

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-293709

(43)Date of publication of application : 20.10.2000

(51)Int.Cl.

G06T 17/00

G06T 11/00

G06T 15/00

(21)Application number : 2000-022197

(71)Applicant : TOSHIBA CORP

(22)Date of filing : 31.01.2000

(72)Inventor : KUNIMATSU ATSUSHI
UENO KYOJI
YASUKAWA HIDEKI
WATANABE YUKIO
KAMEI TAKAYUKI
AMATSUBO TAKANAO

(30)Priority

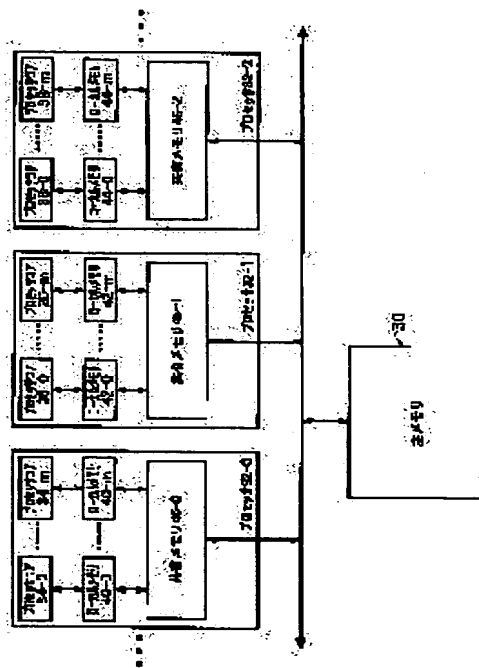
Priority number : 11026563 Priority date : 03.02.1999 Priority country : JP

(54) DEVICE, SYSTEM AND METHOD FOR PROCESSING PICTURE

(57)Abstract:

PROBLEM TO BE SOLVED: To realize a high degree of picture processing with an inexpensive system by improving the capacity of a computer which makes the picture processing.

SOLUTION: A picture processor is composed of a main memory 30 which stores information on three-dimensional objects, a plurality of processor cores 34 (34-0 to 34-m), 36 (36-0 to 36-m), and 38 (38-0 to 38-m), and local memories 40 (40-0 to 40-m), 42 (42-0 to 42-m), and 44 (44-0 to 44-m) and shared memories 46 (46-0 to 46-2) which have a plural tree and are connected between the main memory 30 and processor cores 34, 36, and 38 and successively store a part of the lower-level information at higher levels. The picture processor performs picture processing in parallel by means of the processor cores 34, 36, and 38.



LEGAL STATUS

[Date of request for examination]

28.01.2003

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of

rejection]

[Date of requesting appeal against examiner's
decision of rejection]

[Date of extinction of right]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号
特開2000-293709
(P2000-293709A)

(43) 公開日 平成12年10月20日 (2000. 10. 20)

(51) Int.Cl.⁷

識別記号

F I

テーマコード* (参考)

G 0 6 T 17/00
11/00
15/00

G 0 6 F 15/62
15/72

3 5 0 A
3 5 0
4 5 0 A

審査請求 未請求 請求項の数16 O L (全 31 頁)

(21) 出願番号 特願2000-22197(P2000-22197)

(22) 出願日 平成12年1月31日 (2000. 1. 31)

(31) 優先権主張番号 特願平11-26563

(32) 優先日 平成11年2月3日 (1999. 2. 3)

(33) 優先権主張国 日本 (J P)

(71) 出願人 000003078

株式会社東芝
神奈川県川崎市幸区堀川町72番地

(72) 発明者 国松 敦

神奈川県川崎市幸区小向東芝町1番地 株
式会社東芝マイクロエレクトロニクスセン
ター内

(72) 発明者 上野 喜代治

神奈川県川崎市幸区小向東芝町1番地 株
式会社東芝マイクロエレクトロニクスセン
ター内

(74) 代理人 100083806

弁理士 三好 秀和 (外7名)

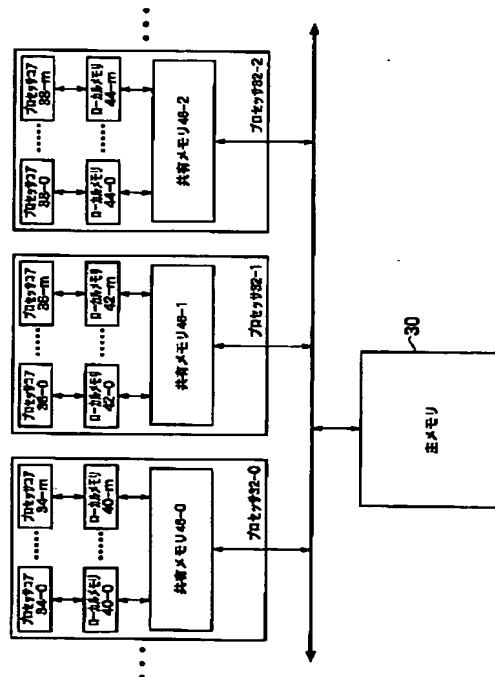
最終頁に続く

(54) 【発明の名称】 画像処理装置、画像処理システムおよび画像処理方法

(57) 【要約】

【課題】 画像処理する計算機の処理能力を向上させ、高度な画像処理を安価なシステムで実現可能な画像処理装置、画像処理システムおよび画像処理方法を提供する。

【解決手段】 3次元物体の情報を格納する主メモリ30、主メモリ30から読み出された3次元物体情報に基づいて画像処理する、複数のプロセッサコア34、36、38、複数の階層を有し、主メモリ30とプロセッサコア34、36、38との間に接続され、下位レベルの情報の一部を上位レベルに順次格納する、ローカルメモリ40、42、44、共有メモリ46、から成る画像処理装置である。この画像処理装置は、複数のプロセッサコア34、36、38による画像処理を並列処理する。



【特許請求の範囲】

【請求項 1】 3次元物体の情報を格納する主記憶部と、
前記主記憶部から読み出された3次元物体情報に基づいて画像処理する、複数の演算部と、
複数の階層を有し、前記主記憶部と前記演算部との間に接続され、下位レベルの情報の一部を上位レベルに順次格納する階層記憶部とを具備し、
前記複数の演算部による画像処理は、並列処理される、
ことを特徴とする画像処理装置。

【請求項 2】 前記階層記憶部の最上位レベルを構成する記憶部は、対応する演算部に従属し、かつ高速アクセス可能な、複数の従属記憶部であり、
前記主記憶部は、前記演算部に共有される、ことを特徴とする請求項 1 に記載の画像処理装置。

【請求項 3】 前記 3次元物体の情報は、前記 3次元物体を包含する空間を、複数の階層で構成された、複数の小空間に分割された状態で、前記主記憶部に格納される、ことを特徴とする請求項 2 に記載の画像処理装置。

【請求項 4】 3次元物体の情報を格納する主記憶部と、前記主記憶部から読み出された3次元物体情報に基づいて画像処理する、複数の演算部と、複数の階層を有し、前記主記憶部と前記演算部との間に接続され、下位レベルの情報の一部を上位レベルに順次格納する階層記憶部と、を有する、複数の画像処理装置と、
該複数の画像処理装置を相互に接続する通信媒体とを具備し、
前記複数の演算部による画像処理は、並列処理される、
ことを特徴とする画像処理システム。

【請求項 5】 前記階層記憶部の最上位レベルを構成する記憶部は、対応する演算部に従属し、かつ高速アクセス可能な、複数の従属記憶部であり、
前記主記憶部は、前記演算部に共有される、ことを特徴とする請求項 4 に記載の画像処理システム。

【請求項 6】 前記 3次元物体の情報は、前記 3次元物体を包含する空間を、複数の階層で構成された、複数の小空間に分割された状態で、前記主記憶部に格納される、ことを特徴とする請求項 5 に記載の画像処理システム。

【請求項 7】 3次元物体の画像処理を分割する工程と、
該分割された複数の処理を、複数の演算部に配分する工程と、
該複数の演算部で、配分された処理を、並列に行なう工程とを少なくとも含み、
前記 3次元物体の情報は、前記 3次元物体を包含する空間を、複数の階層で構成された、複数の小空間に分割された状態で、主記憶部に格納される、ことを特徴とする画像処理方法。

【請求項 8】 前記 3次元物体は、

3次元座標軸および時間軸で定義された4次元空間に存在すると共に、

複数の点で定義された第1の平面図形と、複数の点で定義された、複数の第2の平面図形の組み合わせで定義された第1の立体図形と、少なくとも1つの関数で定義された第2の立体図形、のうちの少なくとも1つである、
ことを特徴とする請求項 3 に記載の画像処理装置。

【請求項 9】 前記 3次元物体は、

3次元座標軸および時間軸で定義された4次元空間に存在すると共に、

複数の点で定義された第1の平面図形と、複数の点で定義された、複数の第2の平面図形の組み合わせで定義された第1の立体図形と、少なくとも1つの関数で定義された第2の立体図形、のうちの少なくとも1つである、
ことを特徴とする請求項 6 に記載の画像処理システム。

【請求項 10】 前記従属記憶部のデータ領域は、3次元空間の階層構造を格納する領域と、3次元物体情報を格納する領域と、に論理的または物理的に分割される、
ことを特徴とする請求項 3 に記載の画像処理装置。

【請求項 11】 前記従属記憶部のデータ領域は、3次元空間の階層構造を格納する領域と、3次元物体情報を格納する領域と、に論理的または物理的に分割される、
ことを特徴とする請求項 6 に記載の画像処理システム。

【請求項 12】 前記複数の演算部間を接続する通信媒体は、複数の演算部が、同一または近傍のアクセス領域にアクセス要求する場合には、それらアクセス要求を結合する、待ち行列手段を備える、ことを特徴とする請求項 1 に記載の画像処理装置。

【請求項 13】 前記複数の演算部間を接続する通信媒体は、複数の演算部が、同一または近傍のアクセス領域にアクセス要求する場合には、それらアクセス要求を結合する、待ち行列手段を備える、ことを特徴とする請求項 4 に記載の画像処理システム。

【請求項 14】 視点から画面上の1つの画素を通過する半直線と、3次元物体と、を交差判定し、該判定の結果から前記画素の画像処理をする画像処理方法において、

3次元空間内の複数の3次元物体の中から1つ選択し、
該選択された3次元物体の情報を取得する第1の工程と、

該取得された情報に基づいて、前記画面上のすべての画素について交差判定する第2の工程と、

前記 3次元空間内のすべての3次元物体に対して、前記第1および第2の工程を実行する第3の工程とを少なくとも含むことを特徴とする画像処理方法。

【請求項 15】 3次元物体の連続画像を生成する画像処理方法において、連続画像を表示する画面の1コマを分割する工程と、

該分割された複数の領域を、複数の演算部に割り当てる工程と、

該複数の演算部で、割り当てられた領域を、並列処理する工程と、

該処理の結果に応じて、前記複数の演算部それぞれの処理時間が等しくなるように、次の1コマを分割する工程とを少なくとも含むことを特徴とする画像処理方法。

【請求項16】 3次元物体の画像を生成する画像処理方法において、

画像を表示する画面を分割する工程と、

該分割された複数の領域を、複数の演算部に割り当てる工程と、

該複数の演算部で、割り当てられた領域を、並列処理する工程とを少なくとも含む、

前記並列処理する工程は、割り当てられた領域の処理が終了した演算部は、他の演算部に割り当てられた未終了の領域を処理する工程を含む、ことを特徴とする画像処理方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、3次元コンピュータグラフィックスを生成する画像処理装置に係り、特に、複数の計算機を用いて並列に画像処理する画像処理装置、この画像処理装置を用いた画像処理システム、およびその画像処理方法に関する。

【0002】

【従来の技術】コンピュータで文字や静止画、動画（アニメーション）を加工する画像処理は、その画像データの大きさから、非常に負荷の大きい処理となる。そのため、レンダリング処理等の画像処理は、スーパーコンピュータやハイエンドワークステーション等の高価な設備を必要とする。

【0003】一方、3次元物体をポリゴンに分割し、ポリゴンを処理単位として描画する画像処理は、安価なシステムで実現可能である。しかしながら、高精細で質感のある映像を描画しようとすれば光の反射や屈折、煙などの粒子の動き等を再現しなければならない。すなわち、物体があたかも3次元空間に存在するように描画するためには、物体の配置（遠くにある物の一部は、近くにある物によって隠れる）や、光線の加減（光源に面した部分は明るく、光源と反対の面は暗い）、物体の材質（鏡面なら光源からの光を反射し、透明の物体なら光線を透過させる）などを考慮し、それらによって物体がどのように見えるかを計算しなければならない。その結果、高価なシステムや膨大な処理時間が結局必要となる。

【0004】

【発明が解決しようとする課題】近年、今まで高価なスーパーコンピュータやワークステーションによらなければ得られなかった高精細かつ質感のある3次元グラフィックスを、安価なシステムで実現することが強く望まれるようになって来ている。

【0005】本発明は、上述の如き事情に鑑みて成されたものであり、その目的は、レンダリング処理の高速化および高効率化を図り、低コストのシステムを実現できる画像処理装置を提供することである。

【0006】本発明の他の目的は、画像処理の高速化、高効率化を実現する、複数の画像処理装置、を含む画像処理システムであって、低コストのシステムを提供することである。

【0007】本発明のさらに他の目的は、レンダリング処理の高速化および高効率化を図り、低コストのシステムを実現できる画像処理方法を提供することである。

【0008】

【課題を解決するための手段】上記課題を解決するため、本発明は、3次元物体の情報を格納する主記憶部と、その主記憶部から読み出された3次元物体情報に基づいて画像処理する、複数の演算部と、複数の階層を有し、主記憶部と演算部との間に接続され、下位レベルの情報の一部を上位レベルに順次格納する階層記憶部、とを少なくとも有する画像処理装置であることを第1の特徴とする。ここで、上記の複数の演算部による画像処理は、並列処理される。

【0009】本発明の第2の特徴は、第1の特徴で述べた画像処理装置において、3次元物体の情報は、その3次元物体を包含する空間を、複数の階層で構成された、複数の小空間に分割された状態で、主記憶部に格納され、その3次元物体は、3次元座標軸および時間軸で定義された4次元空間に存在すると共に、複数の点で定義された第1の平面図形と、複数の点で定義された、複数の第2の平面図形の組み合わせで定義された第1の立体図形と、少なくとも1つの関数で定義された第2の立体図形、のうちの少なくとも1つであることである。

【0010】本発明の第3の特徴は、第1の特徴で述べた画像処理装置において、階層記憶部の最上位レベルを構成する記憶部は、対応する演算部に従属し、かつ高速アクセス可能な、複数の従属記憶部であり、その従属記憶部のデータ領域は、3次元空間の階層構造を格納する領域と、3次元物体情報を格納する領域と、に論理的または物理的に分割される、ことである。また、従属記憶部のデータ領域が、物理的に分割される場合には、3次元物体を格納する領域は、3次元物体情報を保持するメモリ部と、メモリ部中の3次元物体情報の配置を示すテーブル部と、対応する演算部が要求する3次元物体情報が、メモリ部中に存在しない場合、メモリ部の一部と主メモリの一部とを置き換え、要求された3次元物体情報をメモリ部に読み出す制御部とを少なくとも有するように構成すれば良い。

【0011】本発明の第4の特徴は、第1の特徴で述べた画像処理装置において、複数の演算部間を接続する通信媒体は、複数の演算部が、同一または近傍のアクセス領域にアクセス要求する場合には、それらアクセス要求

を結合する、待ち行列手段を備えることである。

【0012】本発明の第5の特徴は、視点から画面上の1つの画素を通過する半直線と、3次元物体と、を交差判定し、その判定の結果から画素の画像処理をする画像処理方法において、3次元空間内の複数の3次元物体の中から1つ選択し、その選択された3次元物体の情報を取得する第1の工程と、その取得された情報に基づいて、画面上のすべての画素について交差判定する第2の工程と、3次元空間内のすべての3次元物体に対して、第1および第2の工程を実行する第3の工程とを少なくとも含むことである。

【0013】本発明の第6の特徴は、3次元物体の連続画像を生成する画像処理方法において、連続画像を表示する画面の1コマを分割する工程と、その分割された複数の領域を、複数の演算部に割り当てる工程と、その複数の演算部で、割り当てられた領域を、並列処理する工程と、その並列処理の結果に応じて、複数の演算部それぞれの処理時間が等しくなるように、次の1コマを分割する工程とを少なくとも含むことである。

【0014】本発明の第7の特徴は、3次元物体の画像を生成する画像処理方法において、画像を表示する画面を分割する工程と、その分割された複数の領域を、複数の演算部に割り当てる工程と、その複数の演算部で、割り当てられた領域を、並列処理する工程とを含むことである。ここで、その並列処理する工程は、割り当てられた領域の処理が終了した演算部は、他の演算部に割り当てられた未終了の領域を処理する工程を含む。

【0015】

【発明の実施の形態】まず最初に、本発明に係るレンダリングおよびそのアルゴリズムであるレイトレーシングについて説明し、次に、本発明に係る画像処理装置、画像処理システム、および画像処理方法について7つの実施の形態を用いて説明する。

【0016】（レンダリングおよびレイトレーシング）本発明に係る画像処理は、3次元グラフィック処理の1つであるレンダリング（rendering）を利用する。そして、そのレンダリングの手法として、本発明に係る画像処理は、処理の並列性が高く、かつ比較的小さいデータで処理が可能なレイトレーシング（ray tracing）というアルゴリズムを使用する。レイトレーシングでは、立方体や球等の物体は数式でモデル化されるため、取り扱われるデータ量は、各オブジェクトをポリゴンで分割する、従来の場合よりも小さくなる。また、視点からの光線を各画素に飛ばし、交差する物体の彩度や明度を計算するので、各画素の計算を独立して行うことが可能となる。

【0017】図1は、本発明に係る画像処理で使用されるレイトレーシングを説明する概念図である。図1に示すように、光源10から出た光は、さまざまなオブジェクト12、14、16に当たり、吸収、反射、屈折され

て観測者の目（視点）18に届く。レイトレーシングでは、各オブジェクト12、14、16の反射や透過などを視点18から所定の画素20を通る光線（レイ）22をさかのぼって追跡する。そして、最初に交差したオブジェクト14をその画素20に描くべきものとし、最終的なイメージを計算する。レイトレーシングでは、各オブジェクト12、14、16の輝度や、透明度、反射等が忠実に再現され、非常にリアルな画像を描写できる。しかしながら、レイトレーシングは、全ての画素についての計算が必要である、オブジェクト数が増加すると計算量が急速に増大する、等の理由から、強力な処理能力が要求されるという問題点を含んでいる。

【0018】そこで、このレイトレーシングと呼ばれるアルゴリズムを、ハードウェアで構成することを考えてみる。図1に示したスクリーン24は、無数の画素20が集まって構成され、さらに、各画素20は、並列に処理可能である。しかしながら、各画素20を並列処理すれば、異なる画素20の計算に同一のオブジェクトデータが同時に必要となる場合が生じ、その結果、メモリ・アクセスの競合を招くおそれがある。本発明は、幾つかの画素20の計算には同じオブジェクトデータが使用されるというデータの局所性（locality）、すなわちメモリ・アクセスの局所性、を利用することにより、各画素20の計算およびメモリ構造の分割・階層化を図るものである。

【0019】本発明においては、まず、図2に示すように、図1のスクリーン24は、複数のサブ・スクリーン26に均等に分割される。さらに各サブ・スクリーン26は、複数のサブ・サブ・スクリーン28に均等に分割される。次に、図3に示すように、各種のオブジェクトデータが格納された主メモリ30と、主メモリ30を共有する複数のプロセッサ32-0、32-1、32-2、…、32-nと、からなるシステムが構成される。プロセッサ32-0は、複数のプロセッサコア34-0、34-1、…、34-mと、プロセッサコア34-0、34-1、…、34-mそれぞれに從属するローカルメモリ40-0、40-1、…、40-mと、ローカルメモリ40-0、…、40-mそれぞれと接続し、プロセッサコア34-0、…、34-mによって共有される共有メモリ46-0と、を有している。プロセッサ32-1は、複数のプロセッサコア36-0、36-1、…、36-mと、プロセッサコア36-0、36-1、…、36-mそれぞれに從属するローカルメモリ42-0、42-1、…、42-mと、ローカルメモリ42-0、…、42-mそれぞれと接続し、プロセッサコア36-0、…、36-mによって共有される共有メモリ46-1と、を有している。プロセッサ32-2は、複数のプロセッサコア38-0、38-1、…、38-mと、プロセッサコア38-0、38-1、…、38-mそれぞれに從属するローカルメモリ44-0、44-

1, ..., 44-mと、ローカルメモリ44-0, ..., 44-mそれぞれと接続し、プロセッサコア38-0, ..., 38-mによって共有される共有メモリ46-2と、を有している。ローカルメモリ40-0, ..., 40-m, 42-0, ..., 42-m, 44-0, ..., 44-mは、主メモリ30および共有メモリ46-0, 46-1, 46-2より、小容量かつ高速アクセス可能である。以下、図示はしないが、プロセッサ32-3, ..., 32-nについても同様である。

【0020】そして、図2の複数のサブ・スクリーン26それぞれが、図3のプロセッサ32-0, 32-1, ..., 32-nそれぞれに割り当てられる。さらに、サブ・スクリーン26を構成するサブ・サブ・スクリーン28それぞれが、各プロセッサ32-0, 32-1, ..., 32-n内のプロセッサコア34-0, ..., 34-m, 36-0, ..., 36-m, 38-0, ..., 38-m, ...それぞれに割り当てられる。たとえば図2のサブ・スクリーン26aが図3のプロセッサ32-1に割り当てられ、サブ・スクリーン26a内のサブ・サブ・スクリーン28aがプロセッサ32-1内のプロセッサコア36-0に割り当てられる。上記のように、各プロセッサ32-0, 32-1, ..., 32-nは共有メモリ46-0, 46-1, ..., 46-nを有しており、この共有メモリ46-0, 46-1, ..., 46-nは各プロセッサ32-1, 32-2, ..., 32-n内の複数のプロセッサコア34-0, ..., 34-m, 36-0, ..., 36-m, 38-0, ..., 38-m, ...によって共有されている。各プロセッサ32-0, 32-1, ..., 32-nは、割り当てられたサブ・スクリーン26の処理で使用されるオブジェクトデータを主メモリ30から読み込む。そして、読み出されたデータを、それぞれの共有メモリ46-0, 46-1, ..., 46-nに格納する。さらに、各プロセッサコア34-0, ..., 34-m, 36-0, ..., 36-m, 38-0, ..., 38-m, ...は、割り当てられたサブ・サブ・スクリーン28の処理で使用されるオブジェクトデータを共有メモリ46-0, 46-1, ..., 46-nから読み込む。そして読み出されたデータを、それぞれのローカルメモリ40-0, ..., 40-m, 42-0, ..., 42-m, 44-0, ..., 44-m, ...に格納する。

【0021】このような構成によれば、各プロセッサ32-0, 32-1, ..., 32-nの、主メモリ30に対するデータ・アクセス回数を抑制できる。それにより、非常に並列度の高い処理を実現できる。結果的には、高度な画像処理を効率的に行うことが可能となる。なお、上記の例では、各プロセッサ32-0, 32-1, 32-2, ..., 32-n内に複数のプロセッサコア34-0, ..., 34-m, 36-0, ..., 36-m, 38-0, ..., 38-m, ...が存在する場合について説明したが、その数は任意である。たとえば1つのプロセッサ

コアであっても構わない。また、共有メモリ46-0, 46-1, 46-2, ..., 46-nは必ずしも必要はない。すなわち、各ローカルメモリ40-0, ..., 40-m, 42-0, ..., 42-m, 44-0, ..., 44-m, ...が主メモリ30に直接接続されていてももちろん構わない。さらに、図1のスクリーン24の分割方法においても、図2に示した例に限定されるわけではなく、最終的にスクリーン24内の幾つかのまとまった画素を1つのプロセッサコア34-0, ..., 34-m, 36-0, ..., 36-m, 38-0, ..., 38-m, ...に割り当てるようにすれば良い。たとえば図2では、各プロセッサコア34-0, ..., 34-m, 36-0, ..., 36-m, 38-0, ..., 38-m, ...に4×4つの画素が割り当てられている。すなわち、各サブ・サブ・スクリーン28は4×4つの画素から構成されている。

【0022】次に、上述したレイトレーシングにOctree（オクツリー）を用いる場合について説明する。オクツリーは、レイトレーシングにおいて、各オブジェクトと光線との交差判定を効率的に行うための探索手法である。具体的には、処理対象となる3次元オブジェクト空間を所定の分割条件に基づいて分割することにより、分割後の空間の大半をオブジェクトが存在しない状態、あるいはその大部分を単一のオブジェクトが占めている状態、とする空間分割手法である。図4（A）および

（B）に、オクツリーの例を示す。図4（A）が、2次元のオクツリーを、図4（B）が、3次元のオクツリーを示している。図4（A）では、2次元空間がx方向に2分割、y方向に2分割され、サブ空間a、b、c、dが形成されている。図4（B）では、3次元空間がx方向に2分割、y方向に2分割、z方向に2分割され、サブ空間a、b、c、d、e、f、g、hが形成されている。各方向の分割の数は上記のように2分割に限られず、それ以外でもよい。以下では、説明の簡単化を図るため、図4（A）に示す2次元空間の場合を用いて説明する。

【0023】図5は、9つのオブジェクト（obj0, obj1, obj2, ..., obj8）が存在する2次元空間に対して、2次元のオクツリーを適用した図である。2次元空間の分割は次の条件に基づいている。

【0024】（1）サブ空間内に3つ以上のオブジェクトが存在すること。

【0025】かつ、

（2）分割階層が2レベル以下であること。

【0026】本発明は、図5の階層的に分割された空間と図3のシステムのメモリ構造との対応づけを行うことにより、効率的なデータ・アクセスを実現する。図6に、図5の階層構造の空間をレイトレーシングに適用した例を示す。レイトレーシングでは、各オブジェクトを数式でモデル化する。したがって、データ量は各オブジェクトの大きさではなく、処理対象の空間内に存在する

オブジェクトの数に比例して増大することになる。ここで、上記のオクツリーを用いることで、各サブ空間を占めるオブジェクトの数をできるだけ少なく、理想的には1つにすることができる。すなわち、各サブ空間に存在するオブジェクトのデータ量は低減されるので、各サブ空間のオブジェクトデータ全体を、図3の各ローカルメモリ40-0, ..., 40-m, 42-0, ..., 42-m, 44-0, ..., 44-m, ...に格納することが可能となる。したがって、図3の各プロセッサコア34-0, ..., 34-m, 36-0, ..., 36-m, 38-0, ..., 38-m, ...は割り当てられた画素の計算に必要なオブジェクトデータを、それぞれのローカルメモリ40-0, ..., 40-m, 42-0, ..., 42-m, 44-0, ..., 44-m, ...にあらかじめ読み込むことができるようになる。さらに、各プロセッサコア34-0, ..., 34-m, 36-0, ..., 36-m, 38-0, ..., 38-m, ...が必要とするデータが、それぞれに従属するローカルメモリ40-0, ..., 40-m, 42-0, ..., 42-m, 44-0, ..., 44-m, ...に存在しない場合、つまりミスヒットの場合には、共有メモリ46-0, 46-1, 46-2, ..., 46-nから、その必要なデータを読み込めば良い。共有メモリ46-0, 46-1, 46-2, ..., 46-nには、上位の空間におけるオブジェクトデータが格納されているからである。本発明によれば、オブジェクトデータの取得を効率的に行い、それにより、交差判定の効率化・高速化を図ることができる。

【0027】(第1の実施の形態)図7は、本発明の第1の実施の形態に係る画像処理システムの構成を示す図である。図7に示すように、本発明の第1の実施の形態に係る画像処理システムは、画像処理を行なう、複数のプロセッサコア48を備え、この複数のプロセッサコア48をネットワーク50で互いに接続した、複数のチップ52を有している。そして、複数のチップ52を備え、この複数のチップ52をネットワーク54で互いに接続した、複数のボード56を有している。さらに、複数のボード56を備え、この複数のボード56をバス

(図示しない)で互いに接続した、複数のコンピュータ58を有している。この複数のコンピュータ58は、ネットワーク60で互いに接続されている。各プロセッサコア48は、それぞれに従属するローカルメモリ(図示しない)を備えている。各チップ52は、それぞれに従属する共有メモリ62を備え、その共有メモリ62は各チップ52内の、複数のプロセッサコア48によって共有される。また、処理対象である3次元空間は、上述したオクツリーによって階層化されている。各共有メモリ62は、階層化されたオブジェクトデータを格納する。

【0028】本発明の第1の実施の形態に係る画像処理システムでは、たとえば画像1コマを描画する場合、その画像1コマは所定の面積ごとに分割される。そして、

分割されたそれぞれのサブ領域の描画処理を各コンピュータ58に割り当てる。各コンピュータ58に割り当てられたサブ領域はさらに分割され、分割されたサブ・サブ領域の描画処理を各ボード56に割り当てる。各ボード56内では、割り当てられたサブ・サブ領域がさらに分割され、分割されたサブ・サブ・サブ領域の描画処理を各プロセッサコア48に割り当てる。このように、本発明の第1の実施の形態に係る画像処理システムでは、画像1コマを小領域、すなわち幾つかの画素の集り、に分割し、各小領域の描画処理を各プロセッサコア48に分配する。そして、各小領域の描画処理を各プロセッサコアが並列に行う。このため、画像1コマ全体についての描画処理を高速かつ効率良く行うことができる。上記の例では、画像1コマを複数のプロセッサコア48で分担して描画処理する場合について記載したが、本発明は、これに限られることはない。たとえば動画の処理を行う場合には、各画像1コマの描画処理を各コンピュータ58に割り当てることも可能である。

【0029】本発明の第1の実施の形態に係る画像処理システムは、たとえば次のようにして画像処理を実行する。図8は、本発明の第1の実施の形態に係る画像処理方法の処理手順を示すフローチャートである。図8に示すように、まず、処理対象である画像を、図2に例示したように、順次分割する(ステップS101)。そして、各プロセッサコア48に、分割された領域の描画処理を分配する(ステップS102)。各プロセッサコア48は、分配された領域の描画処理に必要なオブジェクトデータを主メモリや共有メモリから読み込み、その読み込まれたデータをそれぞれに従属するローカルメモリに格納する(ステップS103)。次に、各プロセッサコア48は、割り当てられた領域の描画処理を行う。この描画処理は、各プロセッサコア48により、並列に行われる(ステップS104)。全プロセッサコア48の描画処理が終了すると(ステップS105YES)、すべての領域の描画結果から処理対象となる画像の生成を行う。生成された画像が、ディスプレイに表示されたり、画像データとして記録媒体に記憶されると(ステップS106)、画像処理が終了する。

【0030】(第2の実施の形態)次に、本発明の第2の実施の形態について説明する。本発明の第2の実施の形態は、処理対象である画像が、静止画像とは異なり、時間とともに内容が変化する動画(アニメーション)である例を示すものである。アニメーションは、一連の連続画像を画面に表示することによって、生成される。このため、レンダリングは、時間が少しずつ異なる複数のフレームそれぞれに対して、実行されることになる。さらに、オクツリーの分割も各フレームごとに行なわなければならない。すなわち、アニメーションにおいては、フレームが変われば3次元オブジェクトの位置、形状、出現、消滅等にも変化が生じる。したがって、フレーム

が変わるたびに空間分割をやり直す必要がある。その結果、フレームの変化ごとに、主メモリ等からローカルメモリにサブ空間情報を転送し直すことが必要となる。

【0031】本発明の第2の実施の形態においては、描画される3次元オブジェクトの情報に時間情報を付加したオブジェクト情報（以下、「4次元オブジェクト情報」と呼ぶ）を生成する。そして、この4次元オブジェクト情報に基づいて、時間方向を考慮して、オクツリーによる空間分割を行うものである。時間情報を考慮して空間分割を行うことで、フレーム変化ごとの空間分割をできるだけ少なくすることが可能となる。それにより、主メモリ等からローカルメモリへの情報転送回数を低減し、処理全体の高速化を図ることができる。以下、本第2の実施の形態について詳細に説明する。

【0032】最初に、本発明の第2の実施の形態に係る4次元オブジェクト情報について説明する。4次元オブジェクト情報は、x座標、y座標、z座標およびt座標で規定される4次元空間で定義される。4次元オブジェクト情報は、周知の3次元オブジェクト情報に時間情報を付加したものである。時間情報の例としてはたとえば次のようなものがある。

【0033】(1) 式

x座標： $x = 3 \cdot t + 4$

y座標： $y = 4 \cdot t + 5$

z座標： $z = 5 \cdot t + 6$

時間の定義域： $0 \leq t \leq 10$

ここで、tは時間を示すパラメータである。以下も同様である。

【0034】(2) 式

$x \cdot x + y \cdot y + z \cdot z + t \cdot t = 10 \cdot 10$

時間の定義域： $-10 \leq t \leq 10$

(3) 数列

$\{t : x\} = \{0 : -1, 1 : 2, 2 : 4\}$

$\{t : y\} = \{0 : 0, 1 : -1, 2 : -2\}$

$\{t : z\} = \{0 : 2, 1 : 2, 2 : 5\}$

ここで、数列によって時間情報を格納した場合、未定義の時間は、線形補間やスプライン補間等の周知のルールによって、補間される。上記の例では、たとえば $t = 0.5$ が未定義の時間である。ただし、補間することなく、すべての時間を数列化することも可能である。

【0035】次に、4次元オブジェクト情報としてはたとえば次のようなものがある。

【0036】(1) 移動する三角形（以下、「obj0」と呼ぶ）

【頂点1】

x座標： $x = 1$

y座標： $y = 1$

z座標： $z = 1 - t$

【頂点2】

x座標： $x = 2$

y座標： $y = 1$

z座標： $z = 1 - t$

【頂点3】

x座標： $x = 1$

y座標： $y = 2$

z座標： $z = 1 - t$

ここで、時間の定義域は $0.5 \leq t \leq 2$ である。

【0037】(2) だんだんと小さくなって消える球（以下、「obj1」と呼ぶ）

$x \cdot x + y \cdot y + z \cdot z + t \cdot t = 1$

ここで、時間の定義域は $0 \leq t \leq 1$ である。

【0038】(3) 移動する球（以下、「obj2」と呼ぶ）

$(x - 1.1) \cdot (x - 1.1) + (y - 1.1) \cdot (y - 1.1) + (z - ((-t/100) - 1.5)) \cdot (z - ((-t/100) - 1.5)) = 0.01 \cdot 0.01$

ここで、時間の定義域は $0 \leq t \leq 2$ である。

【0039】(4) 移動する四面体（以下、「obj3」と呼ぶ）

【頂点1】

x座標： $\{t : x\} = \{-2 : -1, 2 : -1\}$

y座標： $\{t : y\} = \{-2 : -1, 2 : -1\}$

z座標： $\{t : z\} = \{-2 : -1, 2 : -0.5\}$

【頂点2】

x座標： $\{t : x\} = \{-2 : -2, 2 : -2\}$

y座標： $\{t : y\} = \{-2 : -1, 2 : -1\}$

z座標： $\{t : z\} = \{-2 : -1, 2 : -0.5\}$

【頂点3】

x座標： $\{t : x\} = \{-2 : -1, 2 : -1\}$

y座標： $\{t : y\} = \{-2 : -2, 2 : -2\}$

z座標： $\{t : z\} = \{-2 : -1, 2 : -0.5\}$

【頂点4】

x座標： $\{t : x\} = \{-2 : -1, 2 : -1\}$

y座標： $\{t : y\} = \{-2 : -1, 2 : -1\}$

z座標： $\{t : z\} = \{-2 : -2, 2 : -1.5\}$

ここで、時間 $t = -2$ と $t = 2$ との間は、たとえば線形補間を行えばよい。

【0040】次に、上記の4次元オブジェクト(obj0, obj1, obj2, obj3)が存在する4次元空間を空間分割する場合について説明する。図9は、本発明の第2の実施の形態に係る画像処理方法の処理手順を示すフローチャートである。図9に示すように、最初、4次元空間の設定が実行される。4次元空間の設定は、上記の4次元オブジェクトすべてが、その4次元空間に含まれるように、行われる（ステップS201）。上記のオブジェクトがすべて含まれる4次元空間（初期値）としては、たとえば次に示すものがある。

【0041】 $-2 \leq x \leq 2, -2 \leq y \leq 2, -2 \leq z \leq 2, -2 \leq t \leq 2$

次に、設定された4次元空間の空間分割が実行される（ステップS202）。空間分割は、予め定められた分割条件に基づいて実行される。分割条件としてはたとえば次のようなものがある。

【0042】（1）分割対象となる空間に、所定の数以上のオブジェクトが存在すること。

【0043】かつ

（2）最大分割階層レベルを超えていないこと。

【0044】ここでは、「所定の数」を“3”、「最大分割階層レベル」を“3”、とする。初期値として設定された、上記の4次元空間には上記のオブジェクト（obj0, obj1, obj2, obj3）が4つ存在する。当然ながら、分割も行われていない。つまり、分割階層レベルはゼロである。したがって、上記の分割条件は成立し、空間分割が行われる。上記の4次元空間に対して、x軸、y軸、z軸、t軸それぞれについて2分割を行うと、以下のサブ時空間が生成される。

【0045】

サブ時空間0： $0 \leq x \leq 2, 0 \leq y \leq 2, 0 \leq z \leq 2, 0 \leq t \leq 2$

サブ時空間1： $-2 \leq x \leq 0, 0 \leq y \leq 2, 0 \leq z \leq 2, 0 \leq t \leq 2$

サブ時空間2： $0 \leq x \leq 2, -2 \leq y \leq 0, 0 \leq z \leq 2, 0 \leq t \leq 2$

サブ時空間3： $-2 \leq x \leq 0, -2 \leq y \leq 0, 0 \leq z \leq 2, 0 \leq t \leq 2$

サブ時空間4： $0 \leq x \leq 2, 0 \leq y \leq 2, -2 \leq z \leq 0, 0 \leq t \leq 2$

サブ時空間5： $-2 \leq x \leq 0, 0 \leq y \leq 2, -2 \leq z \leq 0, 0 \leq t \leq 2$

サブ時空間6： $0 \leq x \leq 2, -2 \leq y \leq 0, -2 \leq z \leq 0, 0 \leq t \leq 2$

サブ時空間7： $-2 \leq x \leq 0, -2 \leq y \leq 0, -2 \leq z \leq 0, 0 \leq t \leq 2$

サブ時空間8： $0 \leq x \leq 2, 0 \leq y \leq 2, 0 \leq z \leq 2, -2 \leq t \leq 0$

サブ時空間9： $-2 \leq x \leq 0, 0 \leq y \leq 2, 0 \leq z \leq 2, -2 \leq t \leq 0$

サブ時空間10： $0 \leq x \leq 2, -2 \leq y \leq 0, 0 \leq z \leq 2, -2 \leq t \leq 0$

サブ時空間11： $-2 \leq x \leq 0, -2 \leq y \leq 0, 0 \leq z \leq 2, -2 \leq t \leq 0$

サブ時空間12： $0 \leq x \leq 2, 0 \leq y \leq 2, -2 \leq z \leq 0, -2 \leq t \leq 0$

サブ時空間13： $-2 \leq x \leq 0, 0 \leq y \leq 2, -2 \leq z \leq 0, -2 \leq t \leq 0$

サブ時空間14： $0 \leq x \leq 2, -2 \leq y \leq 0, -2 \leq z \leq 0, -2 \leq t \leq 0$

サブ時空間15： $-2 \leq x \leq 0, -2 \leq y \leq 0, -2 \leq z \leq 0, -2 \leq t \leq 0$

生成されたサブ時空間0乃至15のうち、次のサブ時空間がオブジェクト（obj0, obj1, obj2, obj3）のうちの少なくとも1つを包含する。

【0046】

サブ時空間0：obj0, obj1

サブ時空間1：obj1

サブ時空間2：obj1

サブ時空間3：obj1

サブ時空間4：obj0, obj1, obj2

10 サブ時空間5：obj1

サブ時空間6：obj1

サブ時空間7：obj1, obj3

サブ時空間15：obj3

上記から明らかなように、サブ時空間4は、3つのオブジェクトを含んでおり、かつ分割階層レベルは“1”である。すなわち、サブ時空間4に関しては、分割条件が成立する。したがって、サブ時空間4については、さらに空間分割が実行される。サブ時空間4に対して、x軸、y軸、z軸、t軸それぞれについて2分割を行うと、以下のサブ時空間がさらに生成される。

【0047】

サブ時空間4-0： $1 \leq x \leq 2, 1 \leq y \leq 2, -1 \leq z \leq 0, 1 \leq t \leq 2$

サブ時空間4-1： $0 \leq x \leq 1, 1 \leq y \leq 2, -1 \leq z \leq 0, 1 \leq t \leq 2$

サブ時空間4-2： $1 \leq x \leq 2, 0 \leq y \leq 1, -1 \leq z \leq 0, 1 \leq t \leq 2$

サブ時空間4-3： $0 \leq x \leq 1, 0 \leq y \leq 1, -1 \leq z \leq 0, 1 \leq t \leq 2$

30 サブ時空間4-4： $1 \leq x \leq 2, 1 \leq y \leq 2, -2 \leq z \leq -1, 1 \leq t \leq 2$

サブ時空間4-5： $0 \leq x \leq 1, 1 \leq y \leq 2, -2 \leq z \leq -1, 1 \leq t \leq 2$

サブ時空間4-6： $1 \leq x \leq 2, 0 \leq y \leq 1, -2 \leq z \leq -1, 1 \leq t \leq 2$

サブ時空間4-7： $0 \leq x \leq 1, 0 \leq y \leq 1, -2 \leq z \leq -1, 1 \leq t \leq 2$

サブ時空間4-8： $1 \leq x \leq 2, 1 \leq y \leq 2, -1 \leq z \leq 0, 0 \leq t \leq 1$

40 サブ時空間4-9： $0 \leq x \leq 1, 1 \leq y \leq 2, -1 \leq z \leq 0, 0 \leq t \leq 1$

サブ時空間4-10： $1 \leq x \leq 2, 0 \leq y \leq 1, -1 \leq z \leq 0, 0 \leq t \leq 1$

サブ時空間4-11： $0 \leq x \leq 1, 0 \leq y \leq 1, -1 \leq z \leq 0, 0 \leq t \leq 1$

サブ時空間4-12： $1 \leq x \leq 2, 1 \leq y \leq 2, -2 \leq z \leq -1, 0 \leq t \leq 1$

サブ時空間4-13： $0 \leq x \leq 1, 1 \leq y \leq 2, -2 \leq z \leq -1, 0 \leq t \leq 1$

50 サブ時空間4-14： $1 \leq x \leq 2, 0 \leq y \leq 1, -2 \leq$

$$z \leq -1, 0 \leq t \leq 1$$

$$\text{サブ時空間 } 4-15: 0 \leq x \leq 1, 0 \leq y \leq 1, -2 \leq z \leq -1, 0 \leq t \leq 1$$

生成されたサブ時空間 4-0 乃至 4-15 のうち、次のサブ時空間がオブジェクト (obj0, obj1, obj2, obj3) のうち少なくとも 1 つを包含する。

【0048】

サブ時空間 4-0 : obj0

サブ時空間 4-4 : obj2

サブ時空間 4-8 : obj0

サブ時空間 4-11 : obj1

サブ時空間 4-12 : obj2

上記から明らかなように、サブ時空間 4-0, 4-4, 4-8, 4-11, 4-12 のいずれも、1 つのオブジェクトのみを包含する。したがって、分割条件は成立せず、ここで空間分割は終了する。

【0049】次に、生成されたサブ時空間を用いて、レンダリングによる画像処理が実行される (ステップ S203)。レンダリングに用いるアルゴリズムは、レイトレーシングであるとする。図 6 に例示するように、視点とスクリーンが定義される。視点およびスクリーンは、時間と共に変化しないものとする。ここでは、次のように視点とスクリーンが定義される。スクリーンは、4 つの点で定義される。

【0050】(1) 視点

$$(x, y, z) = (1, 1, 4)$$

(2) スクリーン

$$(x, y, z) = (1, 1, 1, 3)$$

$$(x, y, z) = (1, 1, 0, 9, 3)$$

$$(x, y, z) = (0, 9, 0, 9, 3)$$

$$(x, y, z) = (0, 9, 1, 1, 3)$$

次に、定義されたスクリーン上に 1 点 (以下、「A 点」と呼ぶ) が、さらに定義される。そして、視点からこの A 点を通る直線が、通過するサブ時空間が特定される。たとえば A 点が、次の座標を有するとする。

$$\text{【0051】A 点座標: } (x, y, z) = (1, 01, 1, 01, 3)$$

視点と A 点を結ぶ直線は、サブ時空間 0、サブ時空間 4、サブ時空間 8 およびサブ時空間 12 を通過する。したがって、A 点に対応する画素の処理が割り当てられたプロセッサコアは、サブ時空間 0、サブ時空間 4、サブ時空間 8 およびサブ時空間 12 のオブジェクトデータを主メモリ等から読み込み、従属するローカルメモリに格納する。そして、プロセッサコアは、ローカルメモリに格納されたオブジェクトデータを用いて、A 点に対応する画素の描画処理を実行する。

【0052】ここで、上記のサブ時空間は、時間情報を付加したオブジェクト情報に基づいて空間分割されたものである。したがって、視点と A 点を結ぶ直線が通過する、サブ時空間 0、サブ時空間 4、サブ時空間 8 および

サブ時空間 12 は、時間に応じて使用される。すなわち、 $-2 \leq t \leq 0$ では、サブ時空間 8 およびサブ時空間 12 が使用され、 $0 \leq t \leq 2$ では、サブ時空間 0 およびサブ時空間 4 が使用される。たとえば時間単位 “1” を 60 フレームとすれば、 $-2 \leq t \leq 0$ ($60 \times 2 = 120$ フレーム) では、サブ時空間 8 および 12 のオブジェクトデータの転送が行われ、 $0 \leq t \leq 2$ ($60 \times 2 = 120$ フレーム) では、サブ時空間 0 および 4 のオブジェクトデータの転送が実行される。したがって、ローカルメモリへのデータ転送回数は高々 4 回である。一方、フレームごとに空間分割をやり直し、主メモリ等からローカルメモリにサブ空間情報を転送し直すと、データ転送回数は、非常に大きいものとなる。つまり、サブ時空間 0 および 8 がサブ空間 0 に相当し、サブ時空間 4 および 12 がサブ空間 4 に相当する場合、 $-2 \leq t \leq 0$ (120 フレーム) において、サブ空間 0 および 4 のオブジェクトデータの転送が 120 フレームそれぞれについて実行され、 $0 \leq t \leq 2$ (120 フレーム) においても、サブ空間 0 および 4 のオブジェクトデータの転送が 120 フレームそれぞれについて実行される。その結果、ローカルメモリへのデータ転送回数は 480 回となる。

【0053】以上説明したように、本発明の第 2 の実施の形態によれば、アニメーションを表示する場合において、主メモリとローカルメモリ間のデータ転送回数を低減することができる。それにより、画像処理の高速化を図ることが可能となる。また、本発明の第 2 の実施の形態では、次に使用されるサブ時空間を予測することができる。このため、各プロセッサに従属するローカルメモリへのサブ時空間情報のプリロードが可能となり、画像処理のより一層の高速化を実現できる。

【0054】上記の第 2 の実施の形態では、レンダリングに用いるアルゴリズムとしてレイトレーシングを用いて説明したが、本発明はこれに限られるものではない。たとえばアルゴリズムとしてビームトレーシングを用いても良い。ビームトレーシングでは、スクリーンを複数のサブ・スクリーンに分割し、分割されたサブ・スクリーンを各プロセッサに割り当てる。各プロセッサは、割り当てられたサブ・スクリーンの画像処理を行う。図 10 に、図 5 の階層構造の空間をビームトレーシングに適用した例を示す。この例では、スクリーンを 8 つのサブ・スクリーンに分割している。たとえばサブ・スクリーン 3 の処理には、12 個のサブ空間が必要とされる。

【0055】ここで、図 9 のステップ S203 のレンダリングにビームトレーシングが用いられた場合、ビームが通過するサブ空間は次のようになる。視点およびスクリーンの座標は上記と同様である。サブ・スクリーンは、スクリーン上の 4 つの点 (以下、「B 点、C 点、D 点、E 点」と呼ぶ) で定義される。たとえば B 点、C 点、D 点および E 点が、次の座標を有するとする。

【0056】

B点座標: $(x, y, z) = (1.01, 1.01, 3)$

C点座標: $(x, y, z) = (1.01, 1.02, 3)$

D点座標: $(x, y, z) = (1.02, 1.02, 3)$

E点座標: $(x, y, z) = (1.02, 1.01, 3)$

視点とB, C, DおよびE点とを結ぶビームは、サブ時空間0、サブ時空間4、サブ時空間8およびサブ時空間12を通過する。

【0057】本発明の第2の実施の形態では、画像の作成手法としてバウンディングボリュームを用いても良い。レイトレーシングやビームトレーシングでは、オブジェクトが複雑な形状を有する場合や、交差判定に膨大な計算時間を要する場合がある。バウンディングボリュームは、複雑な形状を有するオブジェクトを単純なオブジェクトで包み込み、その単純なオブジェクトに対して交差判定を行う。したがって、計算時間の短縮化が図られる。また、モーションブレンダー手法を利用しても良い。この手法においても、上述したサブ時空間の利用が可能である。このため、レンダリングを用いた場合と同様に、転送回数の低減を図ることができる。

【0058】(第3の実施の形態)次に、本発明の第3の実施の形態について説明する。本発明の第3の実施の形態は、上記の第1および第2の実施の形態のプロセッサコアの構造に関する。上記のオクツリー等の探索木を用いて3次元オブジェクト空間を分割した場合、各プロセッサコアが画像処理する際に必要となるデータは、大別して2つに分類される。すなわち、3次元空間の階層構造に関するデータ(以下、「階層構造データ」と呼ぶ)と、オブジェクトに関するデータ(以下、「オブジェクトデータ」と呼ぶ)、である。第1および第2の実施の形態で述べたように、各プロセッサコアは、割り当てられた領域の画像処理に必要なデータを、それぞれに従属するローカルメモリに記憶する。本発明の第3の実施の形態に係るプロセッサコアは、上記の階層構造データとオブジェクトデータの両方を同一のローカルメモリに記憶せずに、論理的または物理的に独立した異なるローカルメモリに別々に記憶する。

【0059】図11(A)および(B)は、本発明の第3の実施の形態に係るプロセッサの構成を示すブロック図である。図11(A)に示すプロセッサコア64は、1つのデータキャッシュ66を論理的に2つの領域66aおよび66bに分割する。そして、各領域66a、66bに、階層構造データおよびオブジェクトデータそれぞれを記憶する。たとえば領域66aに階層構造データを記憶し、領域66bにオブジェクトデータを記憶する。一方、図11(B)のプロセッサコア68は、階層構造データを記憶する階層構造データキャッシュ70

と、オブジェクトデータを記憶するオブジェクトデータキャッシュ72と、を備える。このプロセッサコア68では、キャッシュ内に格納するデータの特性によって、各キャッシュメモリを最適化できる。たとえばオブジェクトデータは読み出しアクセスだけで書き込みアクセスがないデータである。したがって、オブジェクトデータキャッシュ72をたとえばスクラッチパッドで構成すれば良い。スクラッチパッドは、高速メモリで構成され、データの読み込みを高速に行うことができるものである。したがって、高速にオブジェクトデータの読み込みが可能となり、画像処理の高速化が図られる。

【0060】次に、図11(B)のオブジェクトデータキャッシュ72について詳細に説明する。最初に、一般的なキャッシュの構造について簡単に説明する。キャッシュの構造には、フルアソシアティブ(full associative)方式とダイレクトマップ(direct map)方式がある。図12(A)および(B)に、一般的なキャッシュ構造を示す。図12(A)が、フルアソシアティブ方式のキャッシュ構造であり、図12(B)が、ダイレクトマップ方式のキャッシュ構造である。図12(A)および(B)のキャッシュ構造のいずれにおいても、アドレスのタグとキャッシュ中のタグの比較を行ないながら、データ読み出しの処理が実行される。アドレスのタグとキャッシュ中のタグとが等しく、有効ビットが設定されていれば、読み出しはキャッシュをヒットする。そして、対応するラインのデータが読み出される。そうでない場合は、読み出しはキャッシュをミスし、主メモリから該当するデータがリフィルされ、その後、そのデータが読み出される。いずれのキャッシュ構造であっても、データの最小単位であるラインのサイズは固定されている。そして、データのヒットチェック、リフィル等は、ラインごとに行われる。

【0061】図13に、本発明の第3の実施の形態に係るオブジェクトデータキャッシュに格納されるオブジェクトデータ例を示す。画像処理などの分野においては、データは、オブジェクトと呼ばれる、ある程度大きな単位で、処理される場合がある。図13に示したオブジェクトデータは、球体のオブジェクトに関するデータであり、球に関する半径や、座標、色などの情報の集合である。図13に例示した、オブジェクトデータの特徴としては、次のものが挙げられる。

【0062】(1)1つのオブジェクトに関するデータは、メモリ上にまとまって配置される。

【0063】(2)データに対するアクセスは、1つのオブジェクトを単位として実行される。

【0064】(3)不要なデータは、1つオブジェクトを単位として発生する。

【0065】図13に例示するように、1つのオブジェクトの半径、座標等の属性を示すデータの各アドレスは、そのオブジェクトの先頭アドレスからのオフセット

(以下、「オフセット」と呼ぶ)によって表現することができる。したがって、上述したタグの比較は、オブジェクトの先頭アドレスに対してのみ行えば十分である。しかしながら、上記の図12(A)および(B)に示したキャッシュ構造では、同一のオブジェクトのデータであっても、メモリ・アドレスによって個々に管理されている。このため、オブジェクトデータの読み出しは、オブジェクト単位ではなく、キャッシュのライン単位で、上述したタグの比較が実行されてしまう。このことは、無用なタグ比較を招くこととなり、タグ比較に要するコストを増大させてしまう。

【0066】また、オブジェクトデータの読み出しがミスした場合、メインメモリからそのデータが読み出され、キャッシュのデータ部分に格納される。ところが、このキャッシュの入れ替えは、実際にデータがアクセスされるまでは、実行されない。このため、同一のオブジェクトに関するデータであって、連続的に使用されるデータがミスした場合には、各データのリフィルは、各データアクセスごとに発生してしまう。その結果、演算処理とリフィル処理が、交互に発生し、システムのパフォーマンスを低下させてしまう。

【0067】本発明の第3の実施の形態に係るオブジェクトデータキャッシュは、データの取り扱いを、キャッシュのラインごとではなく、オブジェクトごとに、行なうことを可能とする。それにより、ヒットチェックに要する時間を短縮し、ハードウェアのコストを減少させることができる。また、データを先読みすることで、システムのパフォーマンスを向上させることができる。さらに、不要なオブジェクトデータを一括消去し、データ使用の効率化を図ることもできる。

【0068】図14に、本発明の第3の実施の形態に係るオブジェクトキャッシュの構造を示す。図14に示すように、本発明の第3の実施の形態に係るオブジェクトキャッシュ74は、オブジェクトデータを保持するキャッシュメモリ76と、キャッシュメモリ76中に配置されたオブジェクトデータを示すキャッシュテーブル78と、メインメモリ80に対してデータ転送を指令するオブジェクトトランスファコントローラ82と、を少なくとも備える。

【0069】図15(A)に示すように、キャッシュメモリ76は、各オブジェクトデータを格納するデータフィールドと、そのデータが有効か否かを示すvalid bitフィールドと、そのデータに対応する、キャッシュテーブル78中のエントリNo.を示すエントリNo.フィールドと、からなる、複数のラインを有している。また、キャッシュテーブル78は、上記の図12(A)および(B)に示したキャッシュ構造を採り、図15

(B)に示すように、各オブジェクトデータを格納する、キャッシュメモリ76中のラインのスタートラインNo.を示すスタートラインNo.フィールドと、その

ラインに格納されたオブジェクトの識別番号を示すオブジェクトNo.フィールドと、そのスタートラインNo.が有効か否かを示すvalid bitフィールドと、からなる、複数のエントリを有している。

【0070】次に、本発明の第3の実施の形態に係るオブジェクトキャッシュの動作について図14、図15

(A)および(B)を用いて説明する。図14、図15(A)および(B)では、データ読み出しの際には、CPU84は、オブジェクトの識別番号およびオフセットを、オブジェクトキャッシュ74に指示するものとする。CPU84によるデータアクセス要求があった場合、そのデータがキャッシュメモリ76内に存在すれば(オブジェクトヒット)、キャッシュテーブル78から、そのデータに対応するスタートラインNo.が出力される。出力されたスタートラインNo.と、CPU84からのオフセットとは、加算器86によって加算され、キャッシュメモリ76に出力される。そして、キャッシュメモリ76に格納されたオブジェクトデータがCPU84に出力され、それと同時に、そのデータが有効であることを示すvalid信号もCPU84に渡される。

【0071】一方、CPU84によって要求されたデータがキャッシュメモリ76中に存在しなければ(オブジェクトミス)、valid信号はCPU84に出力されない。CPU84は、オブジェクト識別番号をオブジェクトトランスファコントローラ82に渡し、オブジェクトトランスファコントローラ82にキャッシュメモリ76へのデータ転送を指示する。オブジェクトトランスファコントローラ82は、受け取ったオブジェクト識別番号のオブジェクトデータすべてを、主メモリ80から読み出し、キャッシュメモリ76にリフィルする。このリフィルの際、オブジェクトの各データの大きさが一定であれば、主メモリ80から一定量のデータが転送される。一方、オブジェクトの各データの大きさが異なる場合には、異なる大きさのデータが転送されることになる。このリフィルの作業は、オブジェクトトランスファコントローラ82によって実行され、CPU84の動作とは独立に行われる。各データのリフィルが順次終了次第、CPU84にそのデータおよびvalid信号が出力される。また、その出力されたデータを用いた、CPU84の処理と並列して、その他のデータのリフィルも実行される。

【0072】本発明の第3の実施の形態に係るオブジェクトキャッシュ74では、ヒットチェックが、キャッシュメモリ76の各ラインに対してではなく、キャッシュテーブル78の各エントリに対して実行される。キャッシュテーブル78のエントリの数は、キャッシュメモリ76のラインの数と比べて小さいので、ヒットチェックのコストを低減できる。また、オブジェクトデータ全体がリフィルされるので、同一オブジェクトのデータが連続して利用される場合には、必要なデータのプリロード

が可能となる。このため、データリフィルのペナルティを小さくできる。

【0073】次に、本発明の第3の実施の形態に係るオブジェクトキャッシュの動作をより詳細に説明する。ここでは、図16に示すように、図14のキャッシュメモリ76が、各ラインを配置可能な場所が2つ存在する、2ウェイセットアソシアティブ方式を採り、一方、キャッシュテーブル78が、各エントリの配置場所が固定されない、フルアソシアティブ方式を採用する。キャッシュテーブル78は、CAM (Content Addressable Memory) を用いて、キャッシュメモリ76の各ラインを求め

る。なお、以下では、オブジェクトヒットの場合とオブジェクトミスの場合を分けて説明するが、実際には、これらは並列処理される。

【0074】初期状態では、キャッシュメモリ76およびキャッシュテーブル78のすべてのvalid bitは、ゼロである。そして、CPU84によるデータアクセス要求があると、キャッシュテーブル78は、CPU84からオブジェクト識別番号を受け取る。オブジェクト識別暗号を受け取ったキャッシュテーブル78は、次のものを出力する。

【0075】(1) 入力されたオブジェクト識別番号が、キャッシュテーブル78中に存在することを示すヒット信号

(2) 入力されたオブジェクト識別信号のオブジェクトの各データを格納する、キャッシュメモリ76中のラインのスタートラインNo.

(3) スタートラインNo. が有効であることを示すvalid信号

(4) キャッシュテーブル78中のいずれのエントリがヒットしたかを示すヒットエントリNo.

ここで、上記の(2)乃至(4)は、上記の(1)がセットされている場合のみに有効である。

【0076】オブジェクトヒットの場合、上記の(1)はセットされるので、上記の(2)乃至(4)は有効となる。そして、加算器86によって、(2)のスタートラインNo. とオフセットが加算される。キャッシュメモリ76へのアクセスは、その加算された結果であるアドレスに対して、行われる。アクセスされると、キャッシュメモリ76は、各ウェイから、次の信号を出力する。

【0077】(5) アクセスされたラインが有効であることを示すvalid信号

(6) そのラインがキャッシュテーブル78のどのエントリに対応するかを示すエントリNo.

(7) そのラインに格納されたオブジェクトデータ
ここで、上記の(6)および(7)は、上記の(5)がセットされている場合のみに有効である。

【0078】オブジェクトヒットの場合、いずれかのウェイでは、上記の(4)のヒットエントリNo. と

(6)のエントリNo. とが一致する。そして、一致したウェイの(5)のvalid信号および(7)のオブジェクトデータが選択される。選択されたオブジェクトデータはCPU84に渡される。一方、CPU84に出力されるvalid信号は、上記の(1)、(3)、(4)と各ウェイの(6)との比較結果の論理和、および選択された(5)、の論理積となる。

【0079】次に、オブジェクトミスの場合について図17を用いて説明する。オブジェクト識別番号を受け取ったキャッシュテーブル78は、要求されたオブジェクトのデータが、キャッシュメモリ76中に存在しないことを示すミス信号を、オブジェクトトランスファコントローラ82に、出力する。ミス信号を受け取ったオブジェクトトランスファコントローラ82は、データ転送可能状態であれば、キャッシュテーブル78中のいずれのエントリに書き込みするかを示す情報(refillentry No.)を、内蔵されたレジスタ88に保持する。この情報は、オブジェクトデータの転送が終了するまで保持される。

【0080】また、オブジェクトトランスファコントローラ82は、オブジェクト識別番号を取得し、主メモリ80中における、そのオブジェクトデータの格納場所を特定する。主メモリ80中に、そのオブジェクトデータの開始アドレスやデータサイズが保持されている場合には、オブジェクトトランスファコントローラ82は、それらを取得し、データ転送に備える。

【0081】データ転送開始時、オブジェクトトランスファコントローラ82は、転送開始信号(transfer start)と、キャッシュメモリ76中のスタートラインNo. を、キャッシュテーブル78に、出力する。キャッシュテーブル78は、受け取ったスタートラインNo. を、指定されたエントリのスタートラインNo. フィールドに書き込む。そして、その書き込みを行なったエントリのvalid bitフィールドをvalid bitをセットする。このvalid bitは、各オブジェクトのデータ転送開始時に、順次セットされる。このため、各データがリフィルされ次第、順次CPU84に転送が可能となる。

【0082】また、オブジェクトトランスファコントローラ82は、書き込み信号(writeenable)、書き込みが行われるウェイ番号、書き込みが行われる、キャッシュメモリ76中のラインNo. を出力する。書き込みが行われるラインNo. は、オブジェクトトランスファコントローラ82が管理する。具体的には、ラインNo. は、スタートラインNo. から順に、書き込みが行われる度に、1ずつカウントアップさせることで、決定される。オブジェクトトランスファコントローラ82は、キャッシュメモリ76の各ラインのエントリNo. フィールドに、そのラインに対応するキャッシュテーブル78中のエントリNo. を書き込み、データフィールドに、主メモリから取得したオブジェクトデータを、書

き込む。ウェイの選択は、LRU法などによって行なえば良い。また、転送されるオブジェクトデータによっては、再利用の可能性が、CPU84の演算結果から求められる場合がある。この場合には、オブジェクトトランスファコントローラ82が、CPU84から情報を得ることで、ウェイの選択を実行できる。

【0083】キャッシュメモリ76は、データ書き込みと同時に、各ラインのvalid bitをセットする。すでにvalid bitがセットされているラインについては、そのラインのエントリNo. フィールドに格納されたエントリNo. をキャッシュテーブル78に出力する。キャッシュテーブル78は、受け取ったエントリNo. のエントリのvalid bitをリセットし、そのエントリを無効にする。

【0084】次に、本発明の第3の実施の形態に係るキャッシュメモリ76およびキャッシュテーブル78に対する書き込みが、どのようにして行われるかについて図18、図19(a)乃至(e)を用いて説明する。図18は、主メモリ80中に格納された複数のオブジェクトのデータの例である。図19(a)乃至(e)は、図18のデータがキャッシュメモリ76に書き込まれる際の、キャッシュメモリ76およびキャッシュテーブル78の内容の変化を示す図である。説明の簡単化を図るため、図18の各オブジェクトは、オブジェクト1、オブジェクト2、オブジェクト3、…、オブジェクト8の順でCPU84からアクセスされるものとする。また、図19(a)乃至(e)では、キャッシュメモリ76が、1ウェイ16ラインであり、キャッシュテーブル78が、4エントリであるとする。

【0085】まず、初期化状態では、キャッシュメモリ76およびキャッシュテーブル78のすべてのvalid bitは、リセットされている。この場合、CPU84がオブジェクト1に対してアクセス要求しても、キャッシュテーブル78中には、オブジェクト1に対応するエントリは存在しない(図19(a)参照)。したがって、キャッシュメモリ76の0ラインから順に、オブジェクト1のデータがリフィルされる。また、同時に、このリフィルが、キャッシュテーブル78の0エントリに対応することも、書き込まれる。一方、キャッシュテーブル78には、書き込まれたデータがオブジェクト1のものであることと、そのデータが書き込まれたラインのスタートラインNo. と、が書き込まれる(図19(b)参照)。CPU84がオブジェクト2に対してアクセス要求した場合も同様に行われる(図19(c)参照)。

【0086】次に、CPU84がオブジェクト3に対してアクセス要求した場合、キャッシュテーブル78の2エントリにオブジェクト3に関する情報が書き込まれ、キャッシュメモリ76のライン12から順にオブジェクトデータが書き込まれる(図19(d)参照)。キャッシュメモリ76の15ラインまで書き込まれると、再び

0ラインから書き込みが行われる。この時、0ラインおよび1ラインに対応する、キャッシュテーブル78の0エントリのvalid bitは、リセットされる(図19(e)参照)。

【0087】(第4の実施の形態)次に、本発明の第4の実施の形態について説明する。複数のノードと1つのターゲットがネットワークで相互に接続されているシステムにおいて、複数のノードからターゲットに対して同時にアクセス要求が起きた場合、そのアクセス要求の競合によってシステム全体の性能が低下してしまう場合がある。この性能低下は、図7に示した第1の実施の形態の画像処理システムにおいても、同様に起こり得る。すなわち、図7のチップ52、ボード56、コンピュータ58それぞれの内部においても、アクセス要求の競合が起こり得る。複数のコンピュータ58を接続するネットワーク60においても、もちろん同様である。本発明の第4の実施の形態は、複数のノードから同時に1つのターゲットにアクセス要求が頻繁に生じる画像処理システムに係るものである。すなわち、同一のターゲットに対する、複数のアクセス要求を1つにまとめてバースト転送を行うことで、各ノードからターゲットへのアクセス待ち時間(レイテンシ)を小さくし、システム全体の性能低下を防止するものである。

【0088】以下、本発明の第4の実施の形態について図20に示すシステム構成を用いて説明する。図20は、単一の共有メモリ90とn個のプロセッサ92-0, 92-1, …, 92-n-1とが、ネットワーク94を介して接続された、共有メモリ型マルチプロセッサを示している。もちろん、本発明の適用範囲は、マルチプロセッサに限られるものではない。まず最初に、本発明の第4の実施の形態に係るコンバイン機能(Combining機能)および拡張コンバイン機能(拡張Combining機能)について説明する。

【0089】(コンバイン機能)図21は、図20のプロセッサ92-0からのアクセス要求とプロセッサ92-1からのアクセス要求の両方が、共有メモリ90の「0x1000」番地への読み込みアクセスである場合を示している。この場合、この2つのアクセス要求を1つにまとめ、1つのアクセス要求として取り扱うことで、アクセス競合は避けられる。この2つのアクセス要求を1つにまとめて取り扱う機能を「コンバイン機能」と呼ぶ。このコンバイン機能によって、2つのプロセッサ92-0, 92-1からの読み込みアクセスは1つにまとめられ、その1つの読み込みアクセスに基づいて、共有メモリ90の「0x1000」番地のデータが読み出される。そして、その読み出されたデータは2つのプロセッサ92-0, 92-1それぞれに送られる。このようにコンバイン機能は、同一のアドレスに対するアクセス要求を複数のプロセッサが同時に行った場合に、そのアクセス要求を1つにまとめることで、アクセス要求の競合を回避

し、それにより各プロセッサのアクセス待ち時間を低減できる。

【0090】(拡張コンバイン機能) 図22(A)は、図20のプロセッサ92-0からの1ワード(4バイト)アクセス要求が、共有メモリ90の「0x1000」番地への読み込みアクセス、プロセッサ92-1からの1ワードアクセス要求が、「0x1004」番地への読み込みアクセス、である場合を示している。この場合、2つの読み込み先アドレスが同じではないため、上記のコンバイン機能によっては2つのアクセス要求を1つにまとめることはできない。拡張コンバインは、上記のコンバイン機能とは異なり、図22(B)に示すように、異なるアドレスに対する複数のアクセス要求を、1つにまとめることができるものである。この拡張コンバイン機能は、各ノードからの複数のアクセス要求が互いに近接するアドレスに対して行われることが多いシステムにおいて非常に有効なものである。もちろん、同一アドレスに対する複数のアクセス要求の場合であっても構わない。

【0091】次に、本発明の第4の実施の形態に係る待ち行列(queue)システムについて説明する。本発明の第4の実施の形態に係る待ち行列システムは、上記の拡張コンバイン機能を取り入れたシステムである。図23に、本発明の第4の実施の形態に係る待ち行列システムの構成を示す。この待ち行列システムは、図20に示したマルチプロセッサに適用された例である。図23に示すように、本発明の第4の実施の形態に係る待ち行列システムは、調停部96と、比較部98と、待ち行列部100と、マルチキャスト部102と、を少なくとも備える。以下、それぞれについて説明する。

【0092】(調停部) LSI内部では、高い動作周波数が要求される。このため、図20のn個のプロセッサ92-0, ..., 92-n-1すべてから同時に共有メモリ90にアクセス要求があった場合、1クロック・サイクル内で、それらすべてのアクセス要求を比較し、上記の拡張コンバインを実行することは現実的には困難である。図23の調停部96は、複数のアクセス要求の中から、比較部98に送られるアクセス要求を、1クロック・サイクルごとに決定する。図24に、調停部96の構成を示す。図24に示すように、この調停部96は、マルチプレクサ104と、アクセステーブル106と、を少なくとも備える。マルチプレクサ104は、複数のプロセッサ92-0, ..., 92-n-1からのアクセス要求の中から幾つかを選び出し、比較部98に送る。マルチプレクサ104は、最大n個のアクセス要求の中から、最大k本のアクセス要求を選択できる。アクセステーブル106は、アクセス要求が待ち行列部100に格納されているプロセッサを示す、複数のビットを有している。各ビットは、プロセッサ92-0, ..., 92-n-1それぞれに対応する。そして、各ビットは、待ち行列部100にアクセス要求が格納されると、「1」にセ

ットされ、そのアクセス要求が実行されるまで「1」を保持する。マルチプレクサ104は、アクセステーブル106を参照して、比較部98に送るアクセス要求を選択する。

【0093】ここで、待ち行列部100がブロッキング(blocking)である場合、調停部96は、待ち行列部100にアクセス要求がすでに格納されているプロセッサからの新たなアクセス要求は選択しない。一方、ノンブロッキング(nonblocking)な待ち行列部100の場合、アクセステーブル106に各プロセッサからのアクセス要求を過去何サイクル前に受け付けたか、あるいは各プロセッサからのアクセス要求を過去幾つ受け付けたかを示せばよい。アクセステーブル106を参照してアクセス要求を選択すれば、各プロセッサからのアクセス要求をより公平に調停(アービトレーション)することが可能となる。なお、この場合には、アクセステーブル106は、各プロセッサに複数のビットを割り当てて、各プロセッサからのアクセス要求の状態を示すことになる。

【0094】(比較部) 比較部98は、調停部96によって選択されたアクセス要求と待ち行列部100の各エントリに格納されたアクセス要求とを比較する。図25に、比較部98の構成を示す。比較部98は、まず最初に、調停部96によって選択されたアクセス要求と待ち行列部100の先頭エントリに格納されているアクセス要求と、を比較する。この2つのアクセス要求が拡張コンバイン可能であれば、比較部98は、調停部96で選択されたアクセス要求を待ち行列部100の先頭エントリにそのまま送る。そして、送られたアクセス要求は、先頭エントリに格納されているアクセス要求と拡張コンバインされる。一方、拡張コンバイン不可能である場合には、調停部96によって選択されたアクセス要求を、待ち行列部100の第2番目のエントリに格納されているアクセス要求と比較する。この2つのアクセス要求が拡張コンバイン可能であれば、待ち行列部100に送る。一方、不可能である場合には、待ち行列部100の第3番目のエントリに格納されているアクセス要求と比較する。以下同様にして比較を行い、待ち行列部100のすべてのエントリに格納されているアクセス要求と拡張コンバイン不可能である場合には、調停部96によって選択されたアクセス要求を、待ち行列部100の最後尾のエントリに格納する。これらの比較処理は、処理速度向上のため、パイプライン処理で実行される。

【0095】調停部96は、複数のプロセッサからのアクセス要求の中から幾つかを選び出し、比較部98に出力する。このため、拡張コンバインの可能性が低下してしまうおそれがある。本発明の第4の実施の形態では、調停部96および比較部98の動作周波数をその他の動作周波数の通倍とすることで、一度に受け付けるアクセス要求の数の増大を図っている。このことは、各部が互

いに独立動作しているに基づくものである。

【0096】(待ち行列部)待ち行列部100は、複数のエントリを有しており、各エントリにプロセッサからのアクセス要求を格納する。各エントリに格納されたアクセス要求は、先に格納された順に待ち行列部100から取り出される。図26に、待ち行列部100の各エントリの構成を示す。各エントリは、アクセス要求が読み込み動作であるか書き込み動作であるかを示す、1ビット、アクセス要求のアクセスタイプ(たとえば、バーストモード、ワードモード、ハーフモードおよびバイトモードの4種類)を示す、2ビット、ターゲットアドレスを示す、そのアドレス幅分だけの、ビット、を備える。また、各エントリは、拡張コンパインの結果を記憶するコンパイン情報テーブルを有する。このテーブルの実施方法は種々あるが、たとえば8つのプロセッサから構成され、最大バースト数が8であるシステムの場合、バーストアクセスに関するテーブルは、図27に示すテーブルとなる。このテーブルは、プロセッサ0からアドレス「0x1010」への4word-Burst-readと、プロセッサ1からアドレス「0x1008」への2word-Burst-readと、プロセッサ2およびプロセッサ3からアドレス「0x1000」への2word-Burst-readと、が拡張コンパインされていることを示している。また、ワード/ハーフ/バイトアクセスに関するテーブルは、図28に示すテーブルとなる。このテーブルはプロセッサ2およびプロセッサ5からアドレス「0x1000」へのByte-readと、プロセッサ4からアドレス「0x1002」へのByte-Readと、が拡張コンパインされていることを示している。

【0097】ここで、待ち行列部100は、書き込み動作時のデータを格納するためのバッファメモリを備えている。このバッファメモリは、スタック構造を採用し、その構造に追加された最新のデータを最初に使用するようになっている。そのため、待ち行列部100の各エントリそれぞれに対応してメモリ領域を確保する必要はない。したがって、メモリコスト低減の点から望ましいものである。また、このバッファメモリはスタックポインタを内蔵する。スタックポインタ1、2、3はバッファメモリのどこからデータを取り出すかを指示するレジスタである。具体的にはどのエントリの書き込み要求についてのデータであるかを指示している。図29のバッファメモリでは一番下にあるデータ1が現在アクセス中若しくは最も近未来にアクセスされるデータである。

【0098】(第5の実施の形態)次に、本発明の第5の実施の形態について説明する。レイトレーシングは、各画素ごとにオブジェクトとの交差判定を行ない、交差するオブジェクトの彩度や明度を計算する。このため、交差判定対象のオブジェクトデータの総量が、プロセッサに從属するローカルメモリの容量に比べて大きいと、ローカルメモリに対するオブジェクトデータの入れ替えが頻繁に発生することになる。本発明の第5の実施の形

態は、各オブジェクトごとに各画素の交差判定を行うことで、主メモリから各プロセッサのローカルメモリへのオブジェクトデータの転送を最小限に抑え、それにより画像処理の高速化を図るものである。ここでは、各プロセッサには、画面全体の中の4×4(=16)画素が割り当てられている。また、各プロセッサは、割り当てられた画素の処理の途中の状態を保存するステータステーブルを有している。このステータステーブルは、画素の数だけのエントリを有して。16個の画素を処理するプロセッサのステータステーブルは、16個のエントリを有することになる。各エントリは、オブジェクト識別番号、交点座標、および、視点から交点までの距離、それぞれを格納する3つのフィールドを備える。オブジェクト識別暗号に換えて、オブジェクトデータを格納する主メモリ上の開始アドレスとしても良い。なお、以下では、それぞれを、オブジェクトフィールド、交点座標フィールド、距離フィールドと呼ぶ。

【0099】本発明の第5の実施の形態の動作は次の通りである。図30は、本発明の第5の実施の形態に係る画像処理方法の処理手順を示すフローチャートである。最初に、ステータステーブルが初期化される。すなわち、ステータステーブルの各エントリのオブジェクトフィールド、交点座標フィールドは初期化され、距離フィールドは無限大に設定される(ステップS301)。任意のオブジェクトが選択され、そのオブジェクトデータが主メモリからローカルメモリに読み込まれる(ステップS302)。次に、1つの画素が選択され(ステップS303)、その画素を通る光線と上記のステップS302で選択されたオブジェクトとの交差判定が行われる。交差する場合には、視点からその交点までの距離が求められる。そして、その求められた結果が、距離フィールドの値よりも小さければ、オブジェクトフィールド、交点座標フィールドおよび距離フィールドの書き換えが行われる(ステップS304)。すべての画素について上記のステップS304が実行され(ステップS305YES)、すべてのオブジェクトについて上記のステップS303およびステップS304が実行されると(ステップS306YES)、この動作は終了する。

【0100】この動作終了後、各画素に対応するエントリには、交差判定の結果が示されている。すなわち、いずれのオブジェクトとも交差しない画素の距離フィールドは無限大のままである。一方、いずれかのオブジェクトと交差する画素のオブジェクトフィールドは、その交差するオブジェクト識別番号を、交点座標フィールドは、その交点座標を、距離フィールドは、視点からその交点までの距離を、それぞれ格納しているはずである。

【0101】本発明の第5の実施の形態によれば、主メモリからローカルメモリへのオブジェクトデータの転送は、そのオブジェクトの数だけ行えばよいことになる。したがって、転送回数は大幅に低減され、その結果、画

像処理の高速化が図られることになる。なお、オブジェクトデータをスクリーン上に透視変換することで、各オブジェクトについて交差判定の不要な画素を予め特定することが可能である。この場合には、交差判定自体の回数も低減され、より画像処理の高速化が促進されることになる。

【0102】次に、処理対象の3次元空間が上記のオクツリーにより空間分割されている場合における、本発明の第5の実施の形態について説明する。この場合には、各プロセッサのステータステーブルの各エントリには、上記のオブジェクトフィールド、交点座標フィールド、距離フィールドに加えて、対応する画素の処理が終了したか否かを示す終了フィールド、次に処理されるサブ空間を示すサブ空間フィールド、をさらに備える。この場合の動作は次の通りである。図31は、本発明の第5の実施の形態に係る画像処理方法の処理手順を示すフローチャートであって、処理対象の3次元空間がオクツリーで空間分割されている場合を示すフローチャートである。まず最初に、ステータステーブルが初期化される。すなわち、各エントリのオブジェクトフィールド、交点座標フィールド、サブ空間フィールドが初期化され、終了フィールドが未終了状態、距離フィールドが無限大に設定される(ステップS401)。終了フィールドが未終了状態である画素すべてが選択され(ステップS402)、各画素に対応する光線が次に通過するサブ空間が特定される。その特定されたサブ空間の番号が各画素に対応するエントリのサブ空間フィールドに書き込まれる。ただし、光線が通過するサブ空間の存在しない画素については、その終了フィールドに処理終了が書き込まれる(ステップS403)。

【0103】次に、終了フィールドが未終了状態である画素が1つ選択され(ステップS404)、そのサブ空間フィールドが示すサブ空間に存在するオブジェクトデータが主メモリからローカルメモリに読み込まれる(ステップS405)。そして、選択された画素と読み込まれたオブジェクトとの交差判定が行われる。さらに、そのサブ空間をサブ空間フィールドに示す他の画素がある場合には、その画素についても同様に交差判定が行われる。交差する場合には、視点からその交点までの距離が求められる。そして、その求められた距離が、距離フィールドの値よりも小さければ、オブジェクトフィールド、交点座標フィールドおよび距離フィールドの書き換えが行われる。そして、終了フィールドにその画素の処理が終了したことが書き込まれる。ステップS405で読み込まれたオブジェクトデータが複数ある場合には、すべてのオブジェクトについて同様に行われる(ステップS406)。すべての画素の処理が終了すると(ステップS407YES)、この動作は終了する。

【0104】この動作終了後、各画素のエントリには、交差判定の結果が示される。すなわち、いずれのオブジ

ェクトとも交差しない画素の距離フィールドは無限大のままである。一方、いずれかのオブジェクトと交差する画素のオブジェクトフィールドは、その交差するオブジェクトの番号を、交点座標フィールドは、その交点座標を、距離フィールドは、視点からその交点までの距離を、それぞれ格納しているはずである。

【0105】次に、上記のステータステーブルに対する書き込みが、どのように行われるかについて図32、図33(a)乃至図34(g)を用いて説明する。ここでは、図32に示す2次元平面内のオブジェクトとの交差判定を行う場合について説明する。また、図33(a)乃至図34(g)は、図32のオブジェクトとの交差判定における、ステータステーブルの変化を示す図である。図32では、説明の簡便化を図るため、処理対象の空間を2次元で説明するが、処理対象が3次元空間であっても何ら異なる点はない。図32に示すように、この2次元空間108内には、6つのオブジェクトobj1, obj2, obj3, obj4, obj5, obj6が存在している。そして2次元空間108は、サブ空間S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13に分割されている。スクリーン110上には、4つの画素p0, p1, p2, p3がある。

【0106】ステータステーブルの各エントリが初期化される。具体的には、終了フィールドは、未終了状態(No)に、距離フィールドは、無限大(∞)に、それぞれ設定される。また、その他のフィールド(サブ空間フィールド、オブジェクトフィールド、交点座標フィールド)は、リセットされる(図33(a)参照)。

【0107】最初、すべての画素p0乃至p3は、処理が未終了である。したがって、すべての画素が選択される。そして、各画素を通過する光線が、次に通過するサブ空間が特定される。特定されたサブ空間が、各画素に対応するサブ空間フィールドに、書き込まれる。図32では、各画素を通過する光線はすべて、最初、サブ空間S12を通過する。したがって、各サブ空間フィールドには、「S12」が書き込まれる(図33(b)参照)。

【0108】サブ空間S12内のオブジェクトobj5, obj6のデータが、主メモリからローカルメモリに読み込まれる。そして、各画素の光線とobj5およびobj6との交差判定が行われる。図32では、画素p0の光線がobj5と交差し、画素p2の光線がobj6と交差する。したがって、画素p0のオブジェクトフィールドに、オブジェクトobj5が、交点座標フィールドに、交点座標(x1, y1)が、距離フィールドに、距離d1が、それぞれ書き込まれる。同様に、画素p2のオブジェクトフィールドに、オブジェクトobj6が、交点座標フィールドに、交点座標(x2, y2)が、距離フィールドに、距離d2が、それぞれ書き込まれる。そして画素p0およびp2の両方の終了フィールドに、処理が終了したことが書き込まれる。具体的には処理の終了を意味する「Ye

s] が、書き込まれる (図 33 (c) 参照)。

【0109】未終了状態の画素 p1 および p3 が選択される。そして、各画素の光線が、次に通過するサブ空間が特定される。図 32 では、画素 p1 および p3 の光線は、サブ空間 S12 の次に、サブ空間 S13 を通過する。画素 p1 および p3 のサブ空間フィールドには、「S13」が書き込まれる (図 33 (d) 参照)。

【0110】サブ空間 S13 内のオブジェクト obj6 のデータが、主メモリからローカルメモリに読み込まれる。そして、画素 p1 および p3 の光線と obj6 との交差判定が行なわれる。図 32 では、画素 p1 および p3 の光線は共に、オブジェクト obj6 とは交差しない。したがって、画素 p1 および p3 それぞれのオブジェクトフィールド、交点座標フィールド、距離フィールドは、変化しない (図 33 (d) 参照)。

【0111】再び、未終了状態の画素 p1 および p3 が選択される。そして、各画素の光線が、次に通過するサブ空間が特定される。図 32 では、画素 p1 および p3 の光線は、サブ空間 S13 の次に、サブ空間 S10 を通過する。画素 p1 および p3 のサブ空間フィールドには、「S10」が書き込まれる (図 34 (e) 参照)。

【0112】サブ空間 S10 内のオブジェクト obj3 のデータが、主メモリからローカルメモリに読み込まれる。そして、画素 p1 および p3 の光線と obj3 との交差判定が行なわれる。図 32 では、画素 p1 の光線は、obj3 と交差する。画素 p1 のオブジェクトフィールドに、オブジェクト obj3 が、交点座標フィールドに、交点座標 (x3, y3) が、距離フィールドに、距離 d3 が、それぞれ書き込まれる。そして、終了フィールドに、「Yes」が、書き込まれる。一方、画素 p3 の光線は、オブジェクト obj3 とは交差しない。したがって、画素 p3 のオブジェクトフィールド、交点座標フィールド、距離フィールドは、変化しない (図 34 (f) 参照)。

【0113】以下、同様に処理が行なわれる。図 32 では、画素 p3 の光線は、いずれのサブ空間とは交差しない。したがって、画素 p3 の終了フィールドに、「Yes」が書き込まれる。また、距離フィールドは、無限大 (∞) のままとなる (図 34 (g) 参照)。

【0114】以上説明したように、本発明の第 5 の実施の形態によれば、主メモリからローカルメモリへのデータ転送量を大幅に低減することができる。したがって、画像処理の高速化を図ることができる。たとえば、プロセッサに從属するローカルメモリの容量が s、3 次元空間内のすべてのオブジェクトデータがローカルメモリ容量 s の r 倍、画素の数が n の場合、転送されるデータ量は、 $s \cdot r \cdot n$ となる。なぜなら、従来では、各画素ごとにすべてのオブジェクトデータを順にロードして交差判定を行い、順に輝度を求めていくからである。一方、本発明の第 5 の実施の形態では、すべてのオブジェクト

データは、一度だけ主メモリからローカルメモリに転送されるだけである。したがって、転送されるデータ量は $s \cdot r$ で済む。

【0115】さらに、空間分割によっても、各サブ空間に含まれるオブジェクトのデータが各プロセッサに從属するローカルメモリの容量よりも大きくなってしまう場合、同様に上記方法によりデータ転送量を大幅に低減できる。また、オブジェクトデータの先読みが可能なシステムにおいては、複数の画素の処理が終了するまでに次のオブジェクトデータの転送が終了していればよい。このため、本発明の第 5 の実施の形態によれば、従来と比べて転送のオーバーヘッドが生じにくい。1 つの主メモリを有し、複数のプロセッサを用いて並列に画像処理を行う画像処理システムでは、アクセス要求の競合からデータ転送のオーバーヘッドが大きい。したがって、本発明の第 5 の実施の形態は、このようなシステムに非常に有効である。

【0116】(第 6 の実施の形態) 次に、本発明の第 6 の実施の形態について説明する。アニメーション (動画) の 1 コマを均等な画素数の領域に分割し、各領域を複数のプロセッサで分担して画像処理する場合、各プロセッサに割り当てられた領域内の物体の数や形状の違いにより、各プロセッサの処理時間に大きな差が生じることがある。1 コマ全体の描画に要する時間は、各プロセッサの描画処理の中で最も長いものに依存してしまう。このため、1 つのプロセッサの処理時間が他と比べて極端に長いと、全体としては画像処理の遅延化を招いてしまう。また、先行して処理が終了したプロセッサは他のすべてのプロセッサの処理が終了するまでは何も処理しない状態となる。このことは、プロセッサの利用効率を低下させる。

【0117】本発明の第 6 の実施の形態は、各プロセッサの処理時間をできるだけ均等にすることで、1 コマの描画に要する時間を短縮する。それにより、各プロセッサの利用効率を向上させる。具体的には、この第 6 の実施の形態では、直前の 1 コマの画像処理の際の、各プロセッサの処理時間や演算量を表す情報に基づいて、次の 1 コマの画像処理における、各プロセッサに割り当てる領域の大きさを調整する。そして、領域の大きさの調整によって、各プロセッサの処理時間をできるだけ均等にする。本発明の第 6 の実施の形態は、連続するコマ同士の類似性に着目したものである。

【0118】「各プロセッサの処理時間や演算量を表す情報 (以下、「演算量情報」と呼ぶ)」としては、たとえば各プロセッサの消費処理時間、実行サイクル数、ローカルメモリのミス率等が挙げられる。1 コマの描画においてこの演算情報量の値が他より大きいプロセッサは、連続するコマどうしの類似性により、その次のコマの描画においても大きな値を持つことが予想される。そして、この値が大きいことは、そのプロセッサの処理時

間が他のプロセッサと比べて長いことを意味する。そこで、1コマの描画処理においてこの値が大きいプロセッサに対しては描画処理における分担領域を小さくすることで、次の1コマにおける処理時間の短縮化を図る。ここでは、演算量情報として、各プロセッサの実行サイクル数を用いた場合について説明する。もちろん、消費処理時間、ローカルメモリのミス率等を用いても、何ら変わることはない。また、各プロセッサは、少なくとも1つの画素ブロック（画素の集合）の処理を担当する。各画素ブロックは、 $2 \times 2 (= 4)$ 画素で構成されている。

【0119】1コマの描画処理における、各プロセッサが担当する画素ブロックの数は、直前描画情報に基づいて決定される。「直前描画情報」とは、直前の1コマの*

0000 0000 0000 0000 0000 0000 0000 0000 0011 1011 1001 1010 1100 1010
0000 0000

“1”が立つ最上位ビットは、LSBから30ビット目である。したがって、この場合、ビット縮退した値、つまり演算量情報は、30となる。ここで、実際の実行サイクル数を64ビットで表した場合、ビット縮退した値の最大値は64となる。6ビットのパターンは $2^6 = 64$ 通りある。したがって、それらのビットパターンによって、0から $2^6 - 1$ までの数値を十分表すことができる。

【0122】また、各プロセッサは複数の画素ブロックの処理を行うので、各プロセッサの演算量情報は各画素※

0000 0000 0000 0000 0000 0000 0000 0000 0010 0000 0000 0000 0000 0000
0000 0000

次の画素の演算量情報が29の場合、LSBから29ビット目に1を加算する。

★30

0000 0000 0000 0000 0000 0000 0000 0000 0011 0000 0000 0000 0000 0000
0000 0000

次の画素の演算量情報が29の場合、LSBから29ビット目に1を加算する。

☆

0000 0000 0000 0000 0000 0000 0000 0000 0100 0000 0000 0000 0000 0000
0000 0000

図35に、4つのプロセッサ（プロセッサ0、プロセッサ1、プロセッサ2およびプロセッサ3）で16個の画素ブロックの描画を実行した場合における、各画素の描画に要した実行サイクル数およびそのビット縮退した値の例を示す。この描画処理では、各プロセッサそれぞれに、4つの画素ブロックが割り当てられている。図35では、プロセッサ0の実行サイクル数は16500、プロセッサ1の実行サイクル数は70000、プロセッサ2の実行サイクル数は5500、プロセッサ3の実行サイクル数は7500、である。これらの値から、図35の場合の直前描画情報は、次のようになる。

【0127】（プロセッサ0、プロセッサ1、プロセッサ2、プロセッサ3）=（25088, 114688, 6656, 9728）

* 描画処理における、各プロセッサの演算量情報を表す値の集合である。たとえばプロセッサが4つであれば直前描画情報は4つの整数の組となる。演算量情報として、実行サイクル数を採用した場合、その表示は実際の実行サイクル数をビット縮退したもので行う。データ送信量の低減のためである。具体的には、実行サイクル数を64ビットの2進数の固定小数点数で表現した場合に、“1”が立つ最上位ビットの位置を演算量情報とする。小数点位置は、最下位ビット（least significant bit; LSB）の右に固定される。

【0120】たとえば実行サイクル数が10億である場合、64ビットの2進数の固定小数点数は、次の通りである。

【0121】

※ブロックの処理に要した演算量情報の累積となる。演算量情報は、実行サイクル数をビット縮退したものである。したがって、演算量情報の累積は、1画素の演算量情報に、次の画素の演算量情報が示すビットに、“1”を加算することで行われる。

【0123】たとえば1画素の演算量情報が30の場合、64ビットの2進数の固定小数点数は、次の通りである。

【0124】

★【0125】

☆【0126】

この直前描画情報は、プロセッサ1の実行サイクル数が他と比べて突出して大きいことを示している。つまり、プロセッサ1の描画処理に要した時間が最も長かったことを意味している。そこで、次のコマの描画処理の際にはプロセッサ1に割り当てる画素ブロックの数を減じ、プロセッサ間の実行サイクル数の差を小さくする。具体的には、以下の式から各プロセッサが次のコマで担当する画素ブロックの数を決定する。

【0128】 $y = (16/x) / ((1/A) + (1/B) + (1/C) + (1/D))$

ここで、y：次のコマの描画処理の際に、各プロセッサに割り当てられる画素ブロックの数、x：各プロセッサの演算量情報の値の和、A：プロセッサ0の演算量情報の値、B：プロセッサ1の演算量情報の値、C：プロセ

ッサ2の演算量情報の値、D:プロセッサ3の演算量情報の値、である。

【0129】すなわち、次のコマで、各プロセッサに割り当てられる画素ブロックの数は、各プロセッサの直前のコマの演算量情報の値に反比例して配分される。図35の場合、各プロセッサが次のコマの描画の際に割り当てられる画素ブロックの数は、次のようになる。

【0130】(プロセッサ0, プロセッサ1, プロセッサ2, プロセッサ3) = (2, 11, 0, 46, 7, 97, 5, 45)

ここで、画素ブロックの数は自然数でなければならない。そこで、次の調整方法が用いられる。

【0131】(1) 小数点以下を四捨五入する。

【0132】(2) 1未満の数値は1とする。

【0133】(3) 4つの数値の合計を16とする。

【0134】したがって、図35の場合、各プロセッサが次のコマの描画の際に割り当てられる画素ブロックの数は、結局次のようになる。

【0135】(プロセッサ0, プロセッサ1, プロセッサ2, プロセッサ3) = (2, 1, 8, 5)

たとえば次のコマが直前のコマと同一の画像を描画するものである場合、この結果に基づいて、図36に示すように、各プロセッサに割り当てられる画素ブロックの数を調整すれば良い。この場合、プロセッサ0の実行サイクル数は3500、プロセッサ1の実行サイクル数は40000、プロセッサ2の実行サイクル数は38500、プロセッサ3の実行サイクル数は17500、である。直前のコマの描画の際には最大実行サイクル数はプロセッサ1の70000である。今回のコマの描画では最大実行サイクル数はプロセッサ1の40000となり、30000サイクルが削減される。

【0136】本発明の第6の実施の形態によれば、各プロセッサの描画時間をできるだけ均等にし、画像処理の効率化を図ることができる。

【0137】(第7の実施の形態) 次に、本発明の第7の実施の形態について説明する。各プロセッサの計算時間は、担当領域の大きさおよびその中に存在するオブジェクトの数に比例する。したがって、担当する領域にオブジェクトが多いプロセッサと担当する領域にオブジェクトが少ないプロセッサとでは、その処理時間に差が生じる。そのため、1画像の描画処理時間は、最も処理の遅いプロセッサの処理時間に依存することになる。

【0138】本発明の第7の実施の形態では、担当領域の処理を先に終了したプロセッサが、他のプロセッサが担当する未処理領域を、さらに処理することで、画像処理の効率化を図るものである。以下、本発明の第7の実施の形態について図37(A)乃至(D)、図38

(A)乃至(C)を用いて説明する。図37(A)乃至(D)は、4つのプロセッサ1, 2, 3, 4によって画像処理される画面の例を示す図、図38(A)乃至

(C)は、プロセッサの処理方向の例を示す図である。

【0139】図37(A)は、処理対象となる画面を示している。点線で区切られた領域(以下、「単位領域」と呼ぶ)は、各プロセッサが一度に取り扱うことのできる領域である。単位領域Lの処理に要する時間を1t、単位領域Mの処理に要する時間を4t、単位領域Nの処理に要する時間を8tとする。図37(A)に示すように、各プロセッサそれぞれには、6つの単位領域からなる領域が、割り当てられている。

10 【0140】各プロセッサの単位領域を処理する順番を決定する際に、次の2つの判断基準を用いる。

【0141】(1) プロセッサは基本的には担当領域の端に接しない単位領域に向かって処理を行なうこと。つまり、担当領域の内部から画面中心方向の端に接する単位領域に向かって処理を行うこと。

【0142】(2) プロセッサはできるだけ隣接する単位領域を順に処理していくこと。

【0143】ここで、その処理順番としては、たとえば図38(A)の渦巻状に進む順、図38(B)の放射状に進む順、図38(C)の螺旋状に進む順、あるいはこれらの組み合わせにより進む順がある。

【0144】図37(A)では、各プロセッサが担当する単位領域はすべて、それぞれの担当領域の端に接している。そこで、たとえば図37(B)に示す順で、各プロセッサの処理が進められる。

【0145】図37(C)は、処理開始から時間9tが経過した時点の処理の進捗状況を示す図である。この時点では、プロセッサ2は担当領域の処理を終了している。プロセッサ2は最後に処理を行った単位領域に隣接する、他のプロセッサに割り当てられた領域の処理をさらに進める。ここではプロセッサ2が最後に処理を行った単位領域に隣接する領域はプロセッサ1に割り当てられた領域である。そして、その領域はこの時点では未処理である。したがって、プロセッサ2はプロセッサ1の割り当てられた未処理領域の処理を進める。

【0146】図37(D)は、処理開始から時間13tが経過した時点の処理の状況を示す図である。この時点ですべてのプロセッサの処理を終了し、画面全体の描画処理が終了している。各プロセッサが担当領域のみを処理とした場合、画面全体の描画処理は最も処理時間の長いプロセッサに律束される。最も長い処理時間を要するのはプロセッサ1であり、その処理時間は16tとなる。つまり、本例の場合、時間3tだけ短縮されることがわかる。

【0147】本発明の第7の実施の形態によれば、負荷の軽いプロセッサが負荷の重いプロセッサの処理を補助することにより、各プロセッサの処理時間の平均化が図られる。その結果、画像処理の高速化が実現される。

【0148】

50 【発明の効果】本発明によれば、画像処理する計算機の

処理能力を向上させることができる。このため、高度な画像処理を安価なシステムで実現することができる。

【図面の簡単な説明】

【図 1】本発明に係る画像処理方法で使用されるレイトレーシングを説明する概念図である。

【図 2】図 1 のスクリーン 24 を階層分割した例を示す図である。

【図 3】本発明に係る画像処理装置のハードウェア構成を示すブロック図である。

【図 4】(A) は、2 次元のオクツリーを説明する概念図、(B) は、3 次元のオクツリーを説明する概念図である。

【図 5】9 つのオブジェクト (obj0, obj1, obj2, ..., obj8) が存在する 2 次元空間に対して、2 次元のオクツリーを適用した場合を説明する図である。

【図 6】図 5 に示した階層構造の 2 次元空間をレイトレーシングに適用した場合を説明する図である。

【図 7】本発明の第 1 の実施の形態に係る画像処理システムの構成を示す図である。

【図 8】本発明の第 1 の実施の形態に係る画像処理方法の処理手順を示すフローチャートである。

【図 9】本発明の第 2 の実施の形態に係る画像処理方法の処理手順を示すフローチャートである。

【図 10】図 5 に示した階層構造の 2 次元空間をビームトレーシングに適用した場合を説明する図である。

【図 11】(A) は、本発明の第 3 の実施の形態に係る、論理的に分割された 2 つの領域を有するデータキャッシュを備えた、ローカルメモリの構造を示すブロック図、(B) は、本発明の第 3 の実施の形態に係る、物理的に分割された 2 つの領域を有するデータキャッシュを備えた、ローカルメモリの構造を示すブロック図である。

【図 12】(A) は、一般的なフルアソシアティブ方式のキャッシュ構造を示す図、(B) は、一般的なダイレクトマップ方式のキャッシュ構造を示す図である。

【図 13】本発明の第 3 の実施の形態に係るオブジェクトデータキャッシュに格納されるオブジェクトデータ例を示す。

【図 14】本発明の第 3 の実施の形態に係るオブジェクトデータキャッシュの構造を示す図である。

【図 15】(A) は、図 14 に示したキャッシュメモリ 76 の構成を示す図、(B) は、図 14 に示したキャッシュテーブル 78 の構成を示す図である。

【図 16】本発明の第 3 の実施の形態に係るオブジェクトデータキャッシュの、オブジェクトヒット時における動作を説明する図である。

【図 17】本発明の第 3 の実施の形態に係るオブジェクトデータキャッシュの、オブジェクトミス時における動作を説明する図である。

【図 18】本発明の第 3 の実施の形態に係るオブジェク

トデータキャッシュに格納される、複数のオブジェクトデータ例を示す。

【図 19】(a) 乃至 (e) は、本発明の第 3 の実施の形態に係るオブジェクトデータキャッシュが、図 18 のデータを格納する際における、キャッシュメモリおよびキャッシュテーブルの内容の変化を示す図である。

【図 20】本発明の第 4 の実施の形態を説明する際に用いられる例である、共有メモリ型マルチプロセッサを示す図である。

【図 21】図 20 のプロセッサ 92-0 からのアクセス要求とプロセッサ 92-1 からのアクセス要求の両方が、共有メモリ 90 の「0x1000」番地への読み込みアクセスである場合を示す。

【図 22】(A) は、図 20 のプロセッサ 92-0 からの 1 ワード (4 バイト) アクセス要求が、共有メモリ 90 の「0x1000」番地への読み込みアクセス、プロセッサ 92-1 からの 1 ワードアクセス要求が、「0x1004」番地への読み込むアクセス、である場合であって、この 2 つのアクセス要求がコンバイン不可である場合を示す図、(B) は、図 20 のプロセッサ 92-0 からの 1 ワード (4 バイト) アクセス要求が、共有メモリ 90 の「0x1000」番地への読み込みアクセス、プロセッサ 92-1 からの 1 ワードアクセス要求が、「0x1004」番地への読み込むアクセス、である場合であって、この 2 つのアクセス要求が拡張コンバイン可能である場合を示す図である。

【図 23】本発明の第 4 の実施の形態に係る待ち行列システムの構成を示す図である。

【図 24】図 23 の調停部の構成を示す図である。

【図 25】図 23 の比較部の構成を示す図である。

【図 26】図 23 の待ち行列部の構成を示す図である。

【図 27】図 26 のコンバイン情報テーブルを示す図であって、プロセッサ 0 からアドレス「0x1010」への 4 word-Burst-read と、プロセッサ 1 からアドレス「0x1008」への 2 word-Burst-read と、プロセッサ 2 およびプロセッサ 3 からアドレス「0x1000」への 2 word-Burst-read と、が拡張コンバインされている場合を示す図である。

【図 28】図 26 のコンバイン情報テーブルを示す図であって、プロセッサ 2 およびプロセッサ 5 からアドレス「0x1000」への Byte-read と、プロセッサ 4 からアドレス「0x1002」への Byte-Read と、が拡張コンバインされている場合を示す図である。

【図 29】図 23 の待ち行列部に備えられる、バッファメモリの構成を示す図である。

【図 30】本発明の第 5 の実施の形態に係る画像処理方法の処理手順を示すフローチャートである。

【図 31】本発明の第 5 の実施の形態に係る画像処理方法の処理手順を示すフローチャートであって、処理対象の 3 次元空間がオクツリーで空間分割されている場合を

示すフローチャートである。

【図32】本発明の第5の実施の形態を説明する際に用いられる例である、2次元空間内のオブジェクトとの交差判定を行なう場合を示す図である。

【図33】(a)乃至(d)は、図32の交差判定を行なう際の、ステータステーブルの内容の変化を示す図である(その1)。

【図34】(e)乃至(g)は、図32の交差判定を行なう際の、ステータステーブルの内容の変化を示す図である(その2)。

【図35】本発明の第6の実施の形態を説明する際に用いられる例である、4つのプロセッサ(プロセッサ0、プロセッサ1、プロセッサ2およびプロセッサ3)で16個の画素ブロックの描画を実行した場合における、各画素の描画に要した実行サイクル数およびそのビット縮退した値を示す図である。

【図36】図35の4つのプロセッサに割り当てられる画素ブロックの、調整後の数を示す図である。

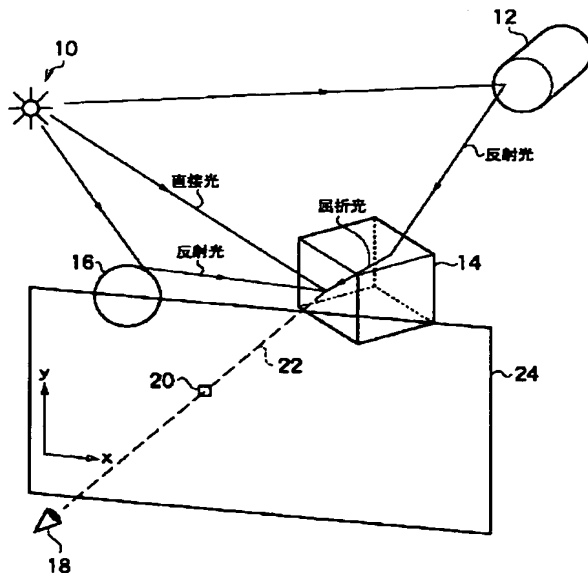
【図37】(A)乃至(D)は、本発明の第7の実施の形態を説明する際に用いられる例である、4つのプロセッサ1, 2, 3, 4によって画像処理される画面の、時間と共に変化する処理状況を示す図である。

【図38】(A)は、図37のプロセッサ1, 2, 3, 4の処理方向が渦巻状である場合を示す図、(B)は、放射状である場合を示す図、(C)は、螺旋状である場合を示す図である。

【符号の説明】

- 10 光源
- 12, 14, 16 オブジェクト
- 18 視点

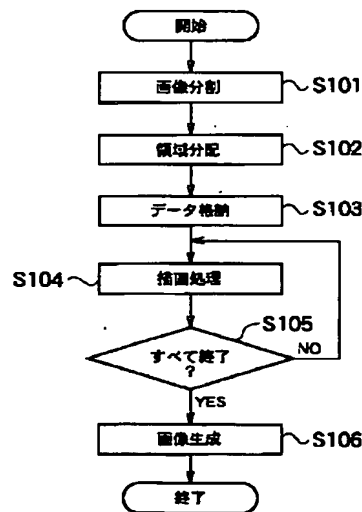
【図1】



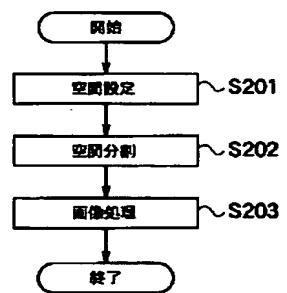
* 20 画素

- 22 光線(レイ)
- 24 スクリーン
- 26 サブ・スクリーン
- 28 サブ・サブ・スクリーン
- 30, 80 主メモリ
- 32, 92 プロセッサ
- 34, 36, 38, 48, 64, 68 プロセッサコア
- 40, 42, 44 ローカルメモリ
- 46, 62, 90 共有メモリ
- 50, 54, 60, 94 ネットワーク
- 52 チップ
- 56 ボード
- 58 コンピュータ
- 66 データキャッシュ
- 70 階層構造データキャッシュ
- 72, 74 オブジェクトデータキャッシュ
- 76 キャッシュメモリ
- 78 キャッシュテーブル
- 82 オブジェクトトランスファコントローラ
- 84 CPU
- 86 加算器
- 88 レジスタ
- 96 調停部
- 98 比較部
- 100 待ち行列部
- 102 マルチキャスト部
- 104 マルチプレクサ
- 106 アクセステーブル
- * 30 108 2次元空間

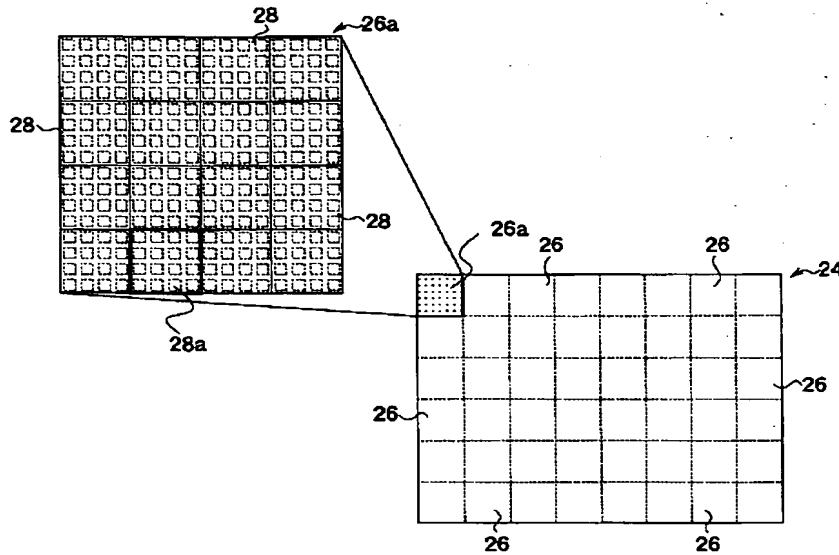
【図8】



【図9】



【図2】



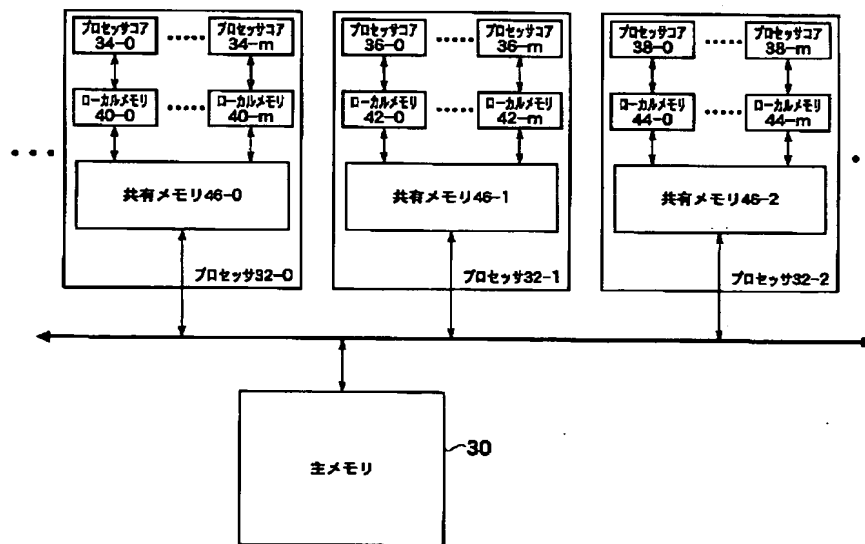
【図13】

ADDRESS	DATA
0x00	sphere
0x01	x coordinate
0x02	y coordinate
0x03	z coordinate
0x04	radius
0x05	color
0x06	ambient
0x07	diffuse
0x08	specular
0x09	-----
0x0a	-----
0x0b	-----
0x0c	-----

【図18】

	start address	size
object1:	0x2000	0x04
object2:	0x2004	0x08
object3:	0x200c	0x06
object4:	0x2002	0x06
object5:	0x200a	0x08
object6:	0x200e	0x04
object7:	0x2012	0x04
object8:	0x2018	0x06

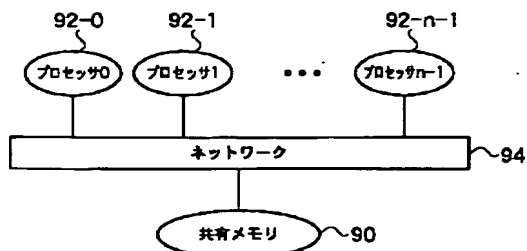
【図3】



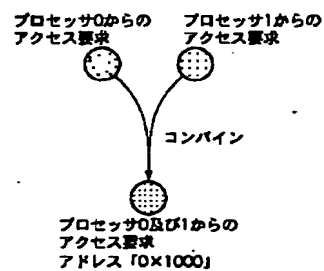
【図28】

	0	1	2	3	4	5	6	7
0								
1								
2	1							
3								
4			1					
5	1							
6								
7								

【図20】



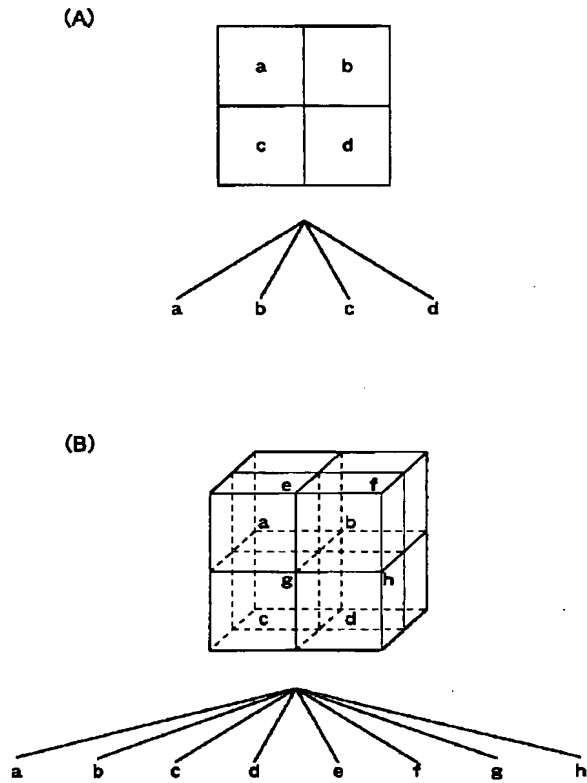
【図21】



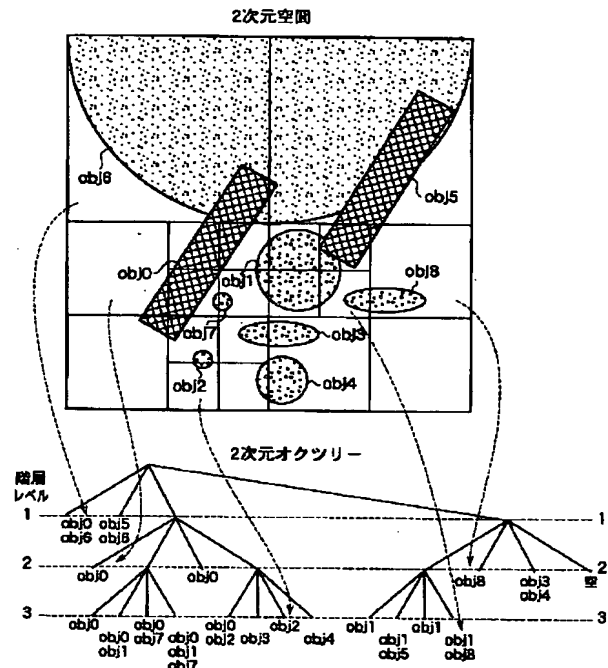
【図27】

	0	1	2	3	4	5	6	7
0				1	1	1	1	1
1			1	1				
2	1	1						
3	1	1						
4								
5								
6								
7								

【図 4】

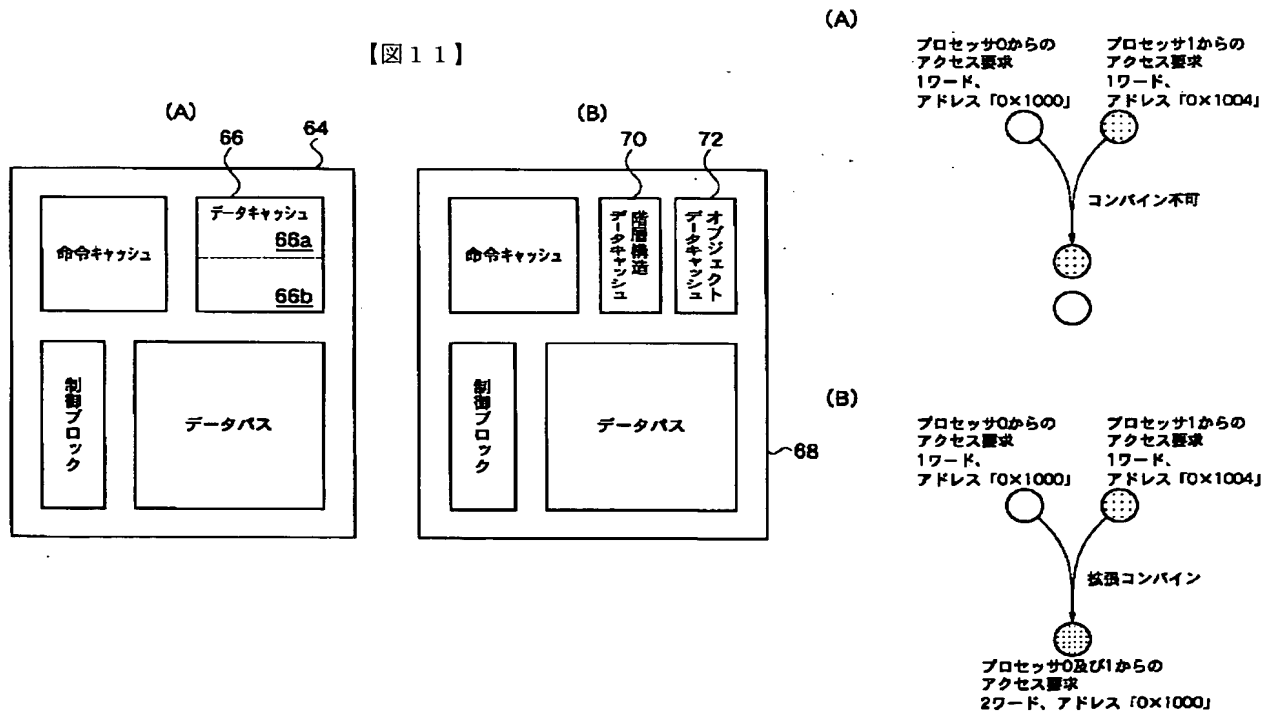


【図 5】

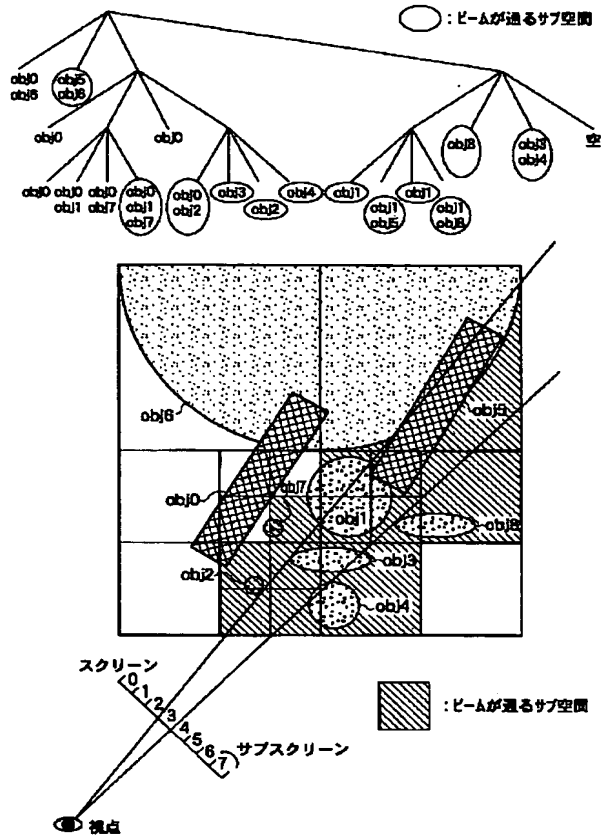


【図 22】

【図 11】



【図10】



【図15】

(A)

Cache Memory

	V	en. No.	DATA
0	0	-	-
1	0	-	-
2	0	-	-
3	0	-	-
4	0	-	-
5	0	-	-
6	0	-	-
7	0	-	-
8	0	-	-
9	0	-	-
10	0	-	-
11	0	-	-
12	0	-	-
13	0	-	-
14	0	-	-
15	0	-	-

← 1 line

V : valid bit
en. No. : entry No.

(B)

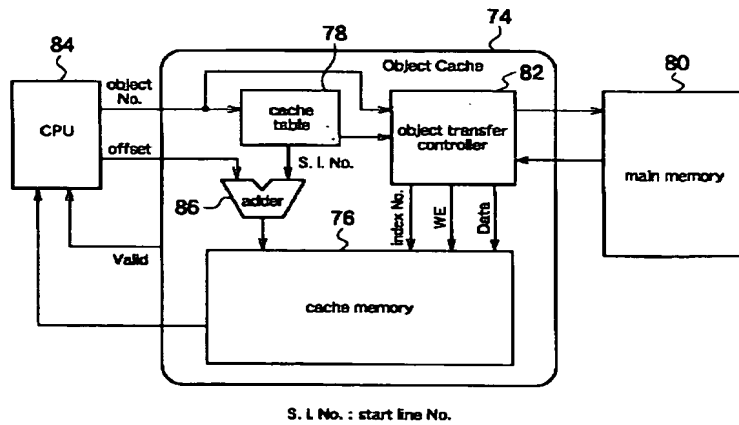
Cache Table

	V	Obj. No.	S. I. No.
0	0	-	-
1	0	-	-
2	0	-	-
3	0	-	-

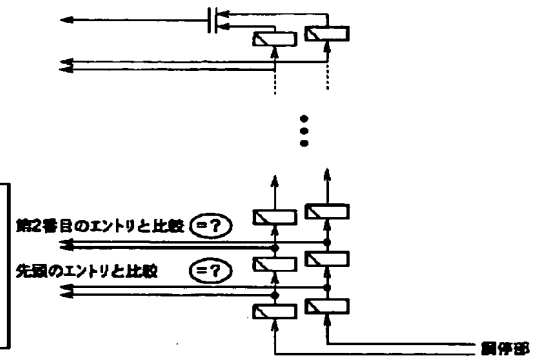
← 1 entry

Obj. No. : Object No.
S. I. No. : start line No.

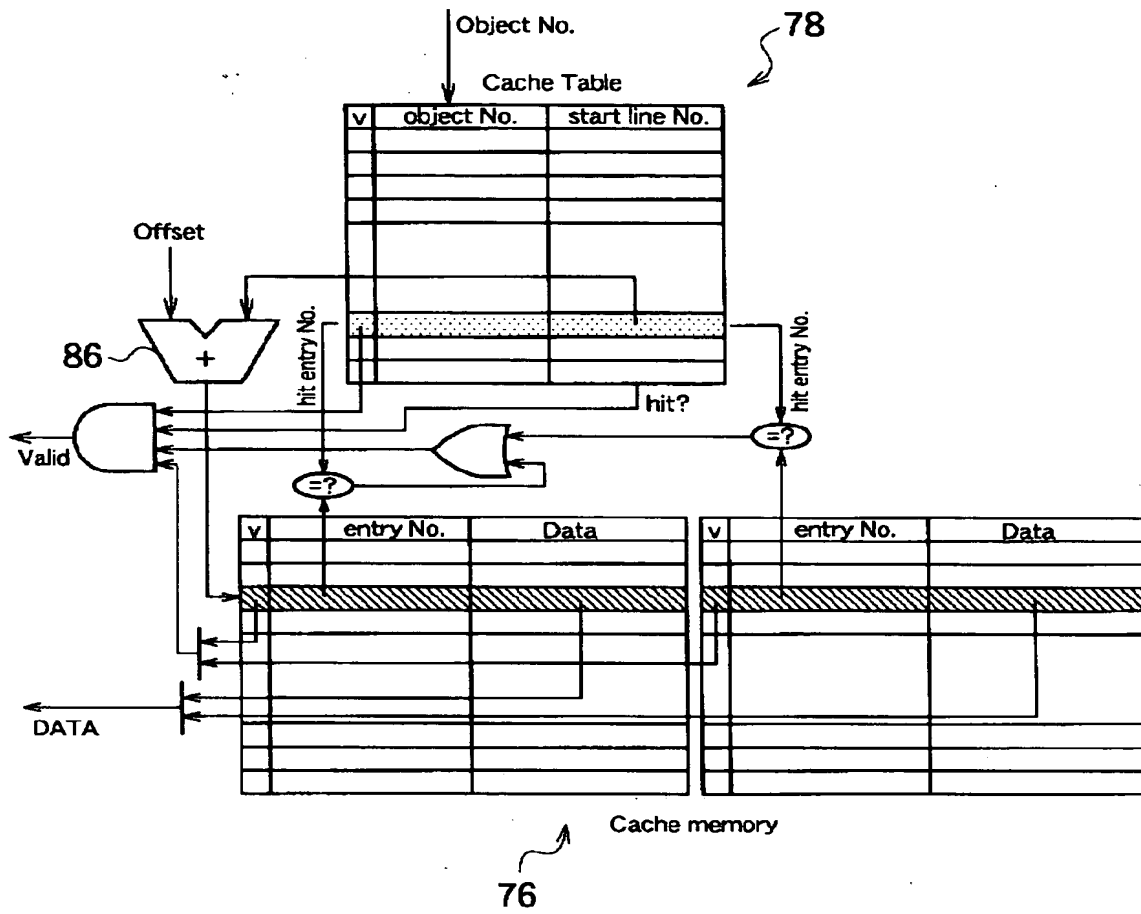
【図14】



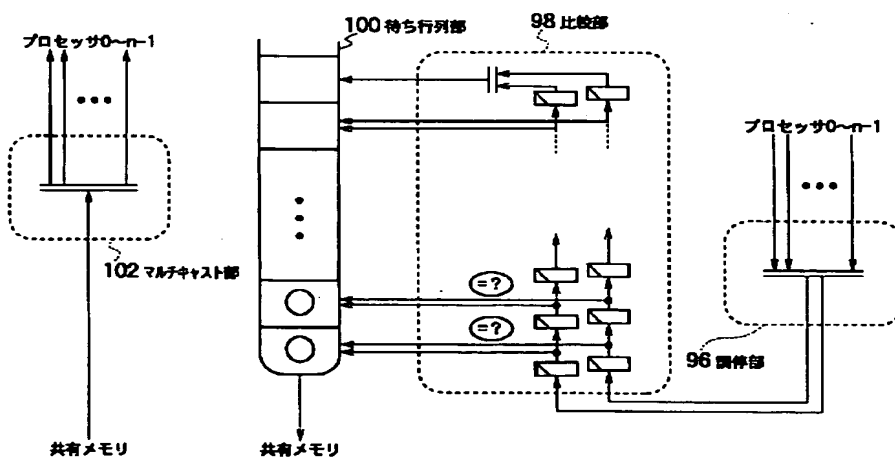
【図25】



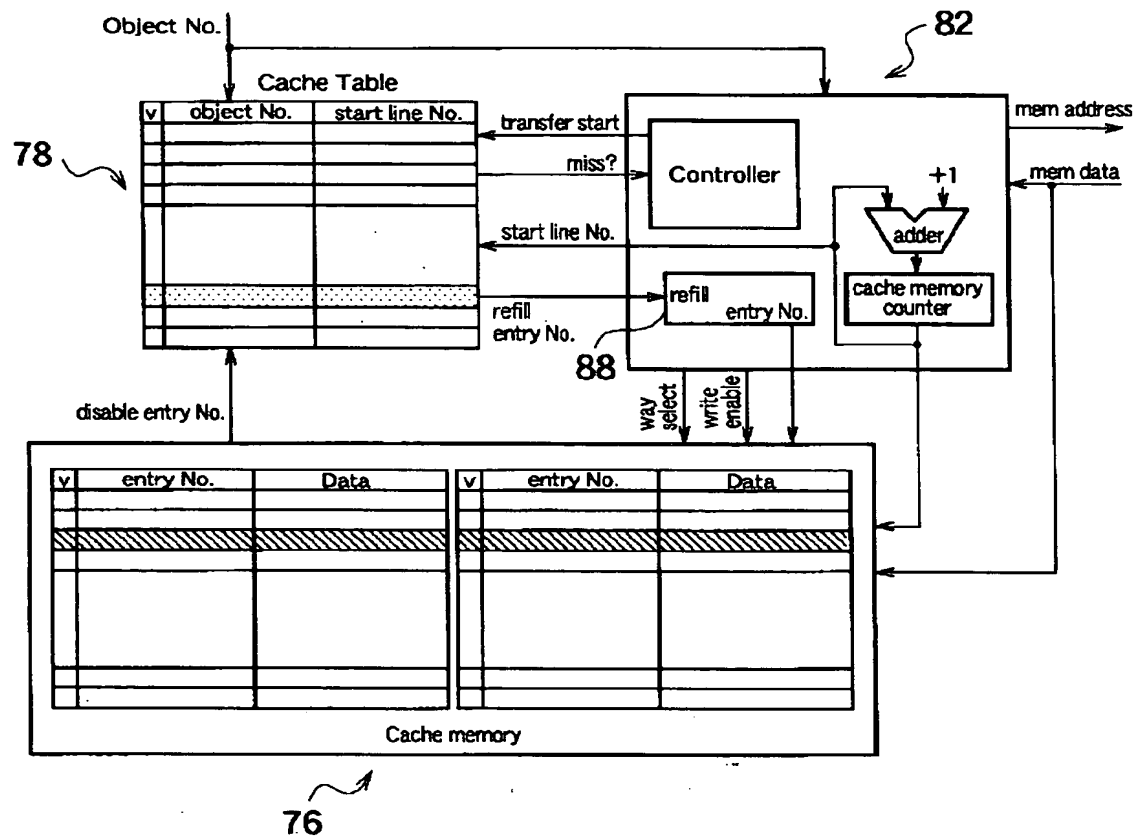
【図16】



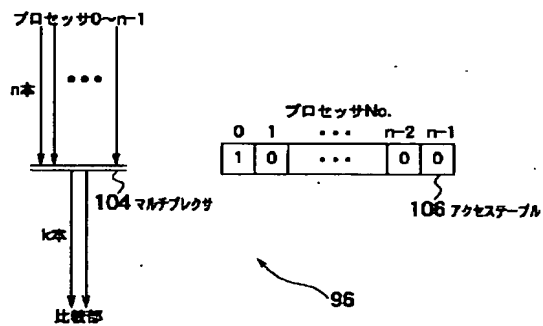
【図23】



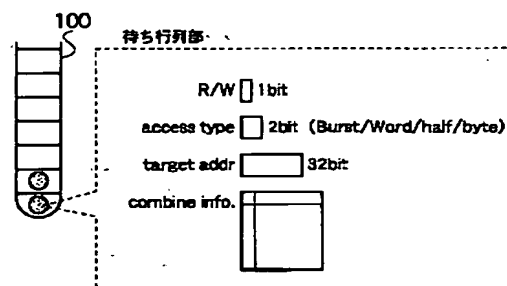
【図17】



【図24】



【図26】



【図 19】

v	obj. No.	S. I. No.
0	0	—
1	0	—
2	0	—
3	0	—

v	obj. No.	S. I. No.
0	1	0
1	0	—
2	0	—
3	0	—

v	obj. No.	S. I. No.
0	1	0
1	1	2
2	0	—
3	0	—

v	en. No.	DATA
0	0	—
1	0	—
2	0	—
3	0	—
4	0	—
5	0	—
6	0	—
7	0	—
8	0	—
9	0	—
10	0	—
11	0	—
12	0	—
13	0	—
14	0	—
15	0	—

v	en. No.	DATA
0	1	obj1-data1
1	1	obj1-data2
2	1	obj1-data3
3	1	obj1-data4
4	0	—
5	0	—
6	0	—
7	0	—
8	0	—
9	0	—
10	0	—
11	0	—
12	0	—
13	0	—
14	0	—
15	0	—

v	en. No.	DATA
0	1	obj1-data1
1	1	obj1-data2
2	1	obj1-data3
3	1	obj1-data4
4	1	obj2-data1
5	1	obj2-data2
6	1	obj2-data3
7	1	obj2-data4
8	1	obj2-data5
9	1	obj2-data6
10	1	obj2-data7
11	1	obj2-data8
12	0	—
13	0	—
14	0	—
15	0	—

(a)

(b)

(c)

v	obj. No.	S. I. No.
0	1	0
1	1	2
2	1	3
3	0	—

v	obj. No.	S. I. No.
0	0	1
1	1	2
2	1	3
3	0	—

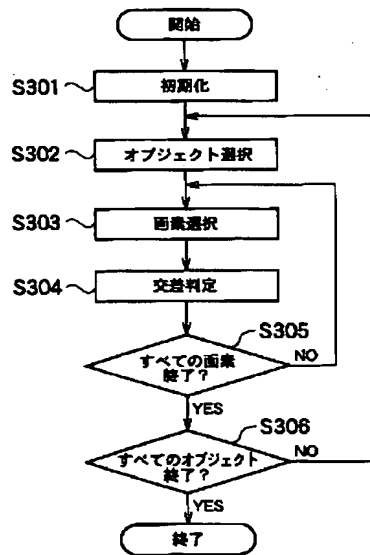
v	en. No.	DATA
0	0	obj1-data1
1	0	obj1-data2
2	0	obj1-data3
3	0	obj1-data4
4	0	obj2-data1
5	0	obj2-data2
6	0	obj2-data3
7	0	obj2-data4
8	0	obj2-data5
9	0	obj2-data6
10	0	obj2-data7
11	0	obj2-data8
12	0	obj3-data1
13	0	obj3-data2
14	0	obj3-data3
15	0	obj3-data4

v	en. No.	DATA
0	1	obj3-data5
1	1	obj3-data6
2	1	obj1-data3
3	1	obj1-data4
4	1	obj2-data1
5	1	obj2-data2
6	1	obj2-data3
7	1	obj2-data4
8	1	obj2-data5
9	1	obj2-data6
10	1	obj2-data7
11	1	obj2-data8
12	1	obj3-data1
13	1	obj3-data2
14	1	obj3-data3
15	1	obj3-data4

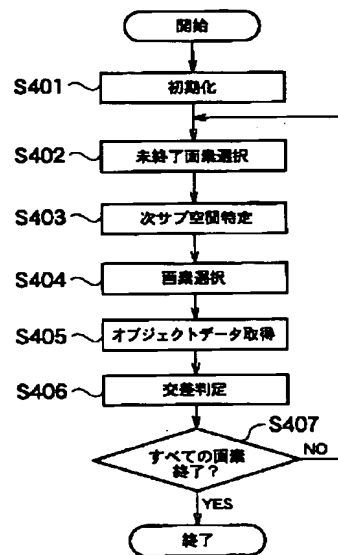
(d)

(e)

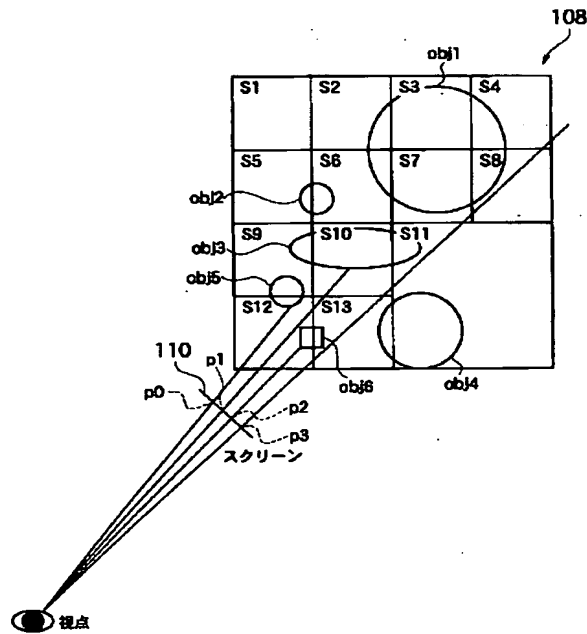
【図30】



【図31】



【図32】



【図33】

(a)

画面No.	終了	サブ空間	オブジェクト	交点座標	距離
p0	No	—	—	—	∞
p1	No	—	—	—	∞
p2	No	—	—	—	∞
p3	No	—	—	—	∞

(b)

画面No.	終了	サブ空間	オブジェクト	交点座標	距離
p0	No	S12	—	—	∞
p1	No	S12	—	—	∞
p2	No	S12	—	—	∞
p3	No	S12	—	—	∞

(c)

画面No.	終了	サブ空間	オブジェクト	交点座標	距離
p0	Yes	S12	obj5	(x1,y1)	d1
p1	No	S12	—	—	∞
p2	Yes	S12	obj8	(x2,y2)	d2
p3	No	S12	—	—	∞

(d)

画面No.	終了	サブ空間	オブジェクト	交点座標	距離
p0	Yes	S12	obj5	(x1,y1)	d1
p1	No	S13	—	—	∞
p2	Yes	S12	obj6	(x2,y2)	d2
p3	No	S13	—	—	∞

【図34】

(e)

画像No.	終了	サブ空間	オブジェクト	交点座標	距離
p0	Yes	S12	obj5	(x1,y1)	d1
p1	No	S10	—	—	∞
p2	Yes	S12	obj8	(x2,y2)	d2
p3	No	S10	—	—	∞

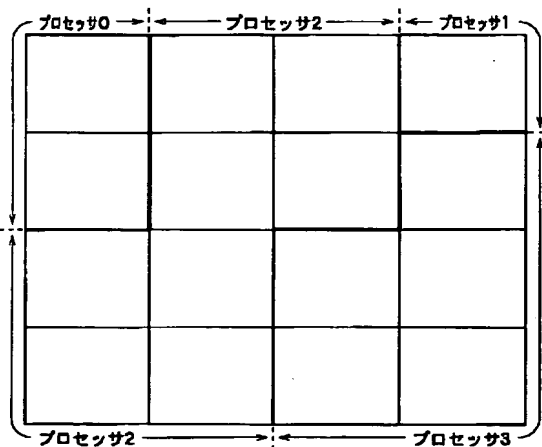
(f)

画像No.	終了	サブ空間	オブジェクト	交点座標	距離
p0	Yes	S12	obj5	(x1,y1)	d1
p1	Yes	S10	obj10	(x3,y3)	d3
p2	Yes	S12	obj8	(x2,y2)	d2
p3	No	S10	—	—	∞

(g)

画像No.	終了	サブ空間	オブジェクト	交点座標	距離
p0	Yes	S12	obj5	(x1,y1)	d1
p1	Yes	S10	obj10	(x3,y3)	d3
p2	Yes	S12	obj8	(x2,y2)	d2
p3	Yes	—	—	—	∞

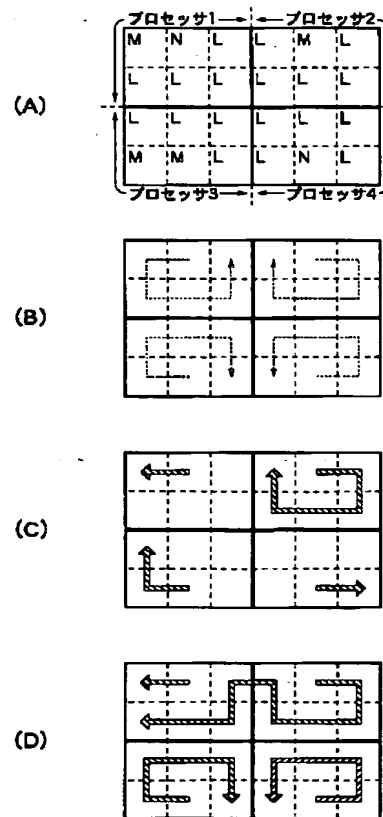
【図36】



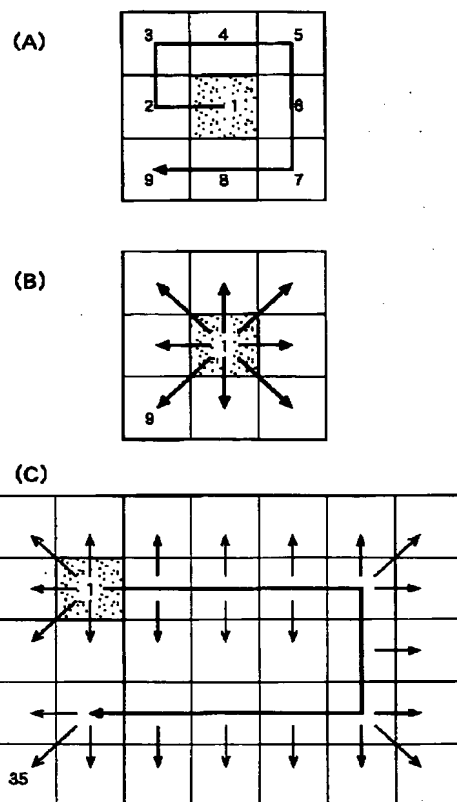
【図35】

プロセッサ0		プロセッサ1	
サイクル数 500 ビット繰返 9	サイクル数 3000 ビット繰返 12	サイクル数 10000 ビット繰返 14	サイクル数 40000 ビット繰返 16
サイクル数 3000 ビット繰返 12	サイクル数 10000 ビット繰返 14	サイクル数 10000 ビット繰返 14	サイクル数 10000 ビット繰返 14
サイクル数 1000 ビット繰返 10	サイクル数 3000 ビット繰返 12	サイクル数 3000 ビット繰返 12	サイクル数 3000 ビット繰返 12
サイクル数 500 ビット繰返 9	サイクル数 1000 ビット繰返 10	サイクル数 1000 ビット繰返 10	サイクル数 500 ビット繰返 9
プロセッサ2		プロセッサ3	

【図37】



【図 38】



フロントページの続き

(72)発明者 安川 英樹
神奈川県川崎市幸区小向東芝町1番地 株
式会社東芝マイクロエレクトロニクスセン
ター内

(72)発明者 渡辺 幸男
神奈川県川崎市幸区小向東芝町1番地 株
式会社東芝マイクロエレクトロニクスセン
ター内

(72)発明者 亀井 貴之
神奈川県川崎市幸区小向東芝町1番地 株
式会社東芝マイクロエレクトロニクスセン
ター内

(72)発明者 雨坪 孝尚
神奈川県川崎市幸区小向東芝町1番地 株
式会社東芝マイクロエレクトロニクスセン
ター内